Extensional and Intensional Strategies

Tony Bourdier   Horatiu Cirstea   Daniel J. Dougherty

INRIA Nancy Grand-Est Nancy Université LORIA
BP 239, 54506 Vandoeuvre-les-Nancy Cedex, France

Worcester Polytechnic Institute
Worcester, MA, 01609, USA

Hélène Kirchner

INRIA Bordeaux Sud-Ouest
351, Cours de la Libération, 33405 Talence, France

This paper is a contribution to the theoretical foundations of strategies. We first present a general definition of abstract strategies which is *extensional* in the sense that a strategy is defined explicitly as a set of derivations of an abstract reduction system. We then move to a more *intensional* definition supporting the abstract view but more operational in the sense that it describes a means for determining such a set. We characterize the class of extensional strategies that can be defined intensionally. We also give some hints towards a logical characterization of intensional strategies and propose a few challenging perspectives.

1   Introduction

Rule-based reasoning is present in many domains of computer science: in formal specifications, rewriting allows prototyping specifications; in theorem proving, rewrite rules are used for dealing with equality, simplifying the formulas and pruning the search space; in programming languages, rules can be explicit like in PROLOG, OBJ or ML, or hidden in the operational semantics; expert systems use rules to describe actions to perform; in constraint logic programming, solvers are described via rules transforming constraint systems. XML document transformations, access-control policies or bio-chemical reactions are a few examples of application domains.

Nevertheless, deterministic rule-based computations or deductions are often not sufficient to capture every computation or proof development. A formal mechanism is needed, for instance, to sequentialize the search for different solutions, to check context conditions, to request user input to instantiate variables, to process sub-goals in a particular order, etc. This is the place where the notion of strategy comes in.

Strategies have been introduced in functional programming (Lisp, ML, Haskell, OBJ), logic programming (PROLOG, CHR), logic-functional languages (Curry, Toy) and constraint programming (CLP). Reduction strategies in term rewriting study which expressions should be selected for evaluation and which rules should be applied. These choices usually increase efficiency of evaluation but may affect fundamental properties of computations such as confluence or (non-)termination. Programming languages like ELAN, Maude and Stratego allow for the explicit definition of the evaluation strategy, whereas languages like Clean, Curry, and Haskell allow for its modification. In theorem proving environments, including automated theorem provers, proof checkers, and logical frameworks, strategies (also called tactics or tacticals in some contexts) are used for various purposes, such as proof search and proof planning, restriction of search spaces, specification of control components, combination of different proof techniques and computation paradigms, or meta-level programming in reasoning systems. Strategies are increasingly useful as a component in systems with computation and deduction explicitly interacting [18, 20]. The complementarity between deduction and computation, as emphasized in particular in

deduction modulo [18], allows us to now envision a completely new generation of proof assistants where customized deductions are performed modulo appropriate and user definable computations [12, 13].

In the fields of system design and verification, *games* —most often two-person path-forming games over graphs— have emerged as a key tool. Such games have been studied since the first half of 20th century in descriptive set theory [30], and they have been adapted and generalized for applications in formal verification; introductions can be found in [29, 8, 44]. Related applications appear in logic [9], planning [40], and synthesis [2]. At first glance the coincidence of the term "strategy" in the domains of rewriting and games appears to be no more than a pun. But it turns out to be surprisingly fruitful to explore the connection and to be guided in the study of the foundations of strategies by some of the insights in the literature of games. This point of view is further developed in this paper.

In order to illustrate the variety of situations involving the notion of strategy, let us give a few examples. In [33], the authors describe a non-deterministic strategy for higher-order rewriting: it amounts to choose an outermost redex and skip redexes that do no contribute to the normal form because they are in a cycle. In proof assistants like Coq [16], tactics are used to describe proof search strategies. For instance, the `orelse` tactic in LCF is defined as follows: given two tactics A and B, apply tactic B only if the application of tactic A either failed or did not modify the proof. In constraint solving, intricate strategies have been defined by combining choice points setting, forward or backward checking, enumeration strategy of values in finite domains, and selection of solutions. Examples can be found in [14]. In game theory, the notion of strategy is crucial to determine the next move for each player [2]. In [17], the idea is applied to computation of normal forms: two players W and B with respective rules R_W and R_B play a game by rewriting terms in the combined signature and we want to know if there exists a winning strategy to reach the normal form.

Strategies are thus ubiquitous in automated deduction and reasoning systems, yet only recently have they been studied in their own right. This paper is a contribution to the concept definition and its theoretical foundation. We try to reconcile different points of view and to compare their expressive power.

In Section 2, we recall the definitions related to abstract reduction systems, before giving in Section 3 the definition of an abstract strategy as a subset of reduction sequences called derivations. In Section 4, we give an intensional definition of strategies compliant with the abstract view but more operational in the sense that it describes a means of building a subset of derivations by associating to a reduction-in-progress the possible next steps in the reduction sequence. Then intensional strategies with memory are defined. This gives the expressive power to build next step with the knowledge of past steps in the derivation. Section 5 explores which abstract strategies can be actually expressed by intensional ones. In order to increase the expressive power of intensional strategies, we eventually propose in Section 6 to define intensional strategies with an accepting condition. Further research questions are presented in Section 7.

2   Abstract reduction systems

When abstracting the notion of strategy, one important preliminary remark is that we need to start from an appropriate notion of *abstract reduction system* (ARS) based on the notion of oriented labeled graph instead of binary relation. This is due to the fact that, speaking of derivations, we need to make a difference between "being in relation" and "being connected". Typically modeling ARS as relations as in [4] allows us to say that, *e.g.*, a and b are in relation but not that there may be several different ways to derive b from a. Consequently, we need to use a more expressive approach, similar to the one proposed in [7, 38] based on a notion of oriented graph. Our definition is similar to the one given in [31] with the

slight difference that we make more precise the definition of steps and labels. Similarly to the step-based definition of an abstract reduction system of [7], this definition that identifies the reduction steps avoids the so-called *syntactic accidents* [36], related to different but indistinguishable derivations.

Definition 1 (Abstract reduction system) *Given a countable set of objects $\mathcal{O}$ and a countable set of labels $\mathcal{L}$ mutually disjoint, an* abstract reduction system *(ARS) is a triple $(\mathcal{O}, \mathcal{L}, \Gamma)$ such that Γ is a functional relation from $\mathcal{O} \times \mathcal{L}$ to $\mathcal{O}$: formally, $\Gamma \subseteq \mathcal{O} \times \mathcal{L} \times \mathcal{O}$ and $(a, \phi, b_1) \in \Gamma$ and $(a, \phi, b_2) \in \Gamma$ implies $b_1 = b_2$.*

The tuples $(a, \phi, b) \in \Gamma$ are called steps *and are often denoted by $a \xrightarrow{\phi} b$. We say that a is the* source *of $a \xrightarrow{\phi} b$, b its* target *and ϕ its* label. *Moreover, two steps are* composable *if the target of the former is the source of the latter.*

Like for graphs and labeled transition systems, in order to support intuition, we often use the obvious graphical representation to denote the corresponding ARS.
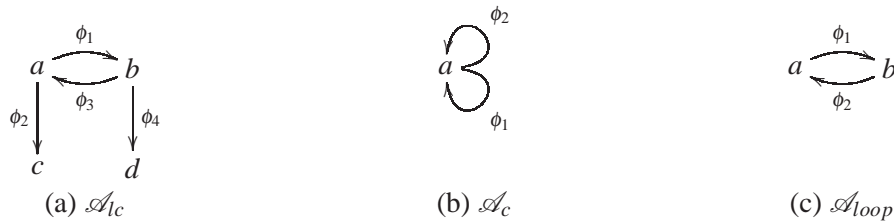

Figure 1: Graphical representation of abstract reduction systems

Example 1 (Abstract reduction systems) *The abstract reduction system*

$$\mathcal{A}_{lc} = (\{a, b, c, d\}, \{\phi_1, \phi_2, \phi_3, \phi_4\}, \{(a, \phi_1, b), (a, \phi_2, c), (b, \phi_3, a), (b, \phi_4, d)\})$$

is depicted in Figure 1(a).

The interest of using the above definition of abstract reduction systems instead of the classical one based on binary relations is illustrated by the abstract reduction system

$$\mathcal{A}_c = (\{a\}, \{\phi_1, \phi_2\}, \{(a, \phi_1, a), (a, \phi_2, a)\})$$

depicted in Figure 1(b).

The "looping" abstract reduction system

$$\mathcal{A}_{loop} = (\{a, b\}, \{\phi_1, \phi_2\}, \{(a, \phi_1, b), (b, \phi_2, a)\})$$

is depicted in Figure 1(c).

The abstract reduction system

$$\mathcal{A}_{nat} = (\{a_i \mid i \in \mathbb{N}\}, \{\phi_i \mid i \in \mathbb{N}\}, \{(a_i, \phi_i, a_{i+1}) \mid i \in \mathbb{N}\})$$
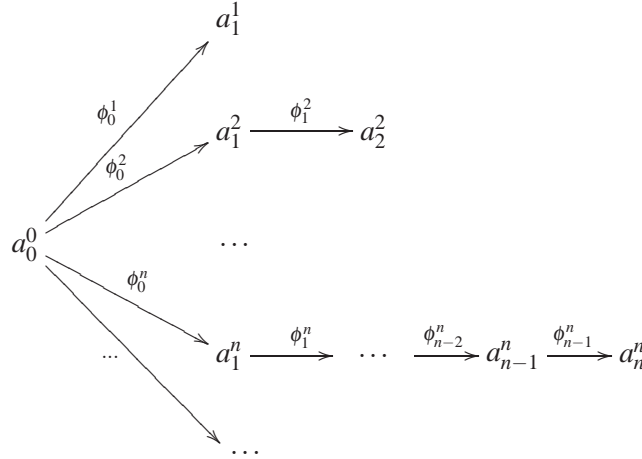
has infinite (but countable) sets of objects and labels, and its relation can be depicted by:

$$a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} \ldots a_i \xrightarrow{\phi_i} a_{i+1} \ldots$$

Another abstract reduction system with an infinite (but countable) set of derivations starting from a same source is

$$\mathscr{A}_{ex} = (\{a_i^j \mid i, j \in \mathbb{N}\}, \{\phi_i^j \mid 0 \leq i < j\}, \{(a_0^0, \phi_0^j, a_1^j) \mid 1 \leq j\} \cup \{(a_i^j, \phi_i^j, a_{i+1}^j) \mid 1 \leq i < j\})$$

whose relation can be depicted by:

The condition that Γ is a functional relation implies that an ARS is a particular case of a labeled transition system. As we will see in what follows, labels characterize the way an object is transformed: given an object and a transformation, there is at most one object resulting from the transformation applied to this particular object. So Definition 1 does not authorize for instance to have $\phi_1 = \phi_2 = \phi$ in $\mathscr{A}_{lc}$ of Example 1.

The next definitions can be seen as a renaming of usual ones in graph theory. Their interest is to allow us to define uniformly derivations and strategies in different contexts.

Definition 2 (Derivation) *Given an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ we call* derivation *over $\mathscr{A}$ any sequence π of steps $\big((t_i, \phi_i, t_{i+1})\big)_{i \in \mathfrak{I}}$ for any right-open interval $\mathfrak{I} \subseteq \mathbb{N}$ starting from 0. If $\mathfrak{I}$ contains at least one element, then:*

- *$Dom(\pi) = t_0$ is called the* source *(or domain) of π,*
- *$l(\pi) = (\phi_i)_{i \in \mathfrak{I}}$ is a sequence called* label *of π,*
- *For any non empty (possibly right-open) subinterval $\mathfrak{I}' \subseteq \mathfrak{I}$, $\pi' = \big((t_i, \phi_i, t_{i+1})\big)_{i \in \mathfrak{I}'}$ is a factor of π. If $\mathfrak{I}'$ contains 0, then π' is a prefix of π. If $\mathfrak{I}' \neq \mathfrak{I}$, π' is a strict factor (or prefix) of π.*

If $\mathfrak{I}$ is finite, it has a smallest upper bound denoted by $n_{\mathfrak{I}}$ or simply n and then:

- *$Im(\pi) = t_n$ is called the* target *(or image) of π,*
- *$|\pi| = card(\mathfrak{I})$ is called the* length *of π,*

In such a case, π is said to be finite and is also denoted by $\pi = (t_0, l(\pi), t_n)$ or $t_0 \xrightarrow{l(\pi)} t_n$. The sequence containing no step is called empty derivation *and is denoted by Λ_Γ and by convention $l(\Lambda_\Gamma) = \varepsilon$ where ε is the empty sequence of elements of $\mathscr{L}$.*

Note that a step may be considered as a derivation of length 1.

We denote by $\Gamma_{\mathscr{A}}^\omega$ (resp. $\Gamma_{\mathscr{A}}^*$, resp. $\Gamma_{\mathscr{A}}^+$) the set of all derivations (resp. finite, resp. non-empty and finite) over $\mathscr{A}$.

Definition 3 (Composable derivations) *Two derivations of the form* $\pi_1 = \left((t_i, \phi_i, t_{i+1})\right)_{i \in \mathfrak{I}_1}$ *and* $\pi_2 = \left((u_i, \phi_i, u_{i+1})\right)_{i \in \mathfrak{I}_2}$ *over a same abstract reduction system* $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ *are* composable *iff either one of the derivations is empty or* $\mathfrak{I}_1$ *is finite and then* $t_{n_1} = u_0$ *where* n_1 *is the smallest upper bound of* $\mathfrak{I}_1$. *In such a case, the composition of* π_1 *and* π_2 *is the unique derivation* $\pi = \left((v_i, \phi_i, v_{i+1})\right)_{i \in \mathfrak{I}}$ *denoted by* $\pi = \pi_1 \pi_2$ *such that for all* $j < |\pi_1|$, $v_j = t_j$ *and for all* $j \geq |\pi_1|$, $v_j = u_{j-|\pi_1|}$.

The composition is associative and has a neutral element which is Λ_Γ. Adopting the product notation, we denote $\prod_{i=1}^{n} \pi_i = \pi_1 \ldots \pi_n$, $\pi^n = \prod_{i=1}^{n} \pi$ and $\pi^\omega = \prod_{i \in \mathbb{N}} \pi$.

Example 2 (Derivations) *Following the previous examples, we have:*

1. $\Gamma_{\mathscr{A}_{lc}}^\omega$ *contains for instance* $\pi_1, \pi_1 \pi_3, \pi_1 \pi_4, \pi_1 \pi_3 \pi_1, (\pi_1 \pi_3)^n, (\pi_1 \pi_3)^\omega, \ldots$, *with* $\pi_1 = (a, \phi_1, b)$, $\pi_2 = (a, \phi_2, c)$, $\pi_3 = (b, \phi_3, a)$, $\pi_4 = (b, \phi_4, d)$;

2. $\Gamma_{\mathscr{A}_c}^+ = \left\{ \prod_{i=1}^{k} \pi_1^{n_i} \pi_2^{m_i} \mid k \geq 1, (n_i, m_i) \in \mathbb{N}^2, (n_i + m_i) > 0 \right\}$ *with* $\pi_1 = (a, \phi_1, a)$ *and* $\pi_2 = (a, \phi_2, a)$.

3. $\Gamma_{\mathscr{A}_{loop}}^+ = \left\{ (\pi_1 \pi_2)^n \pi_1, (\pi_2 \pi_1)^m \pi_2, (\pi_1 \pi_2)^{n+1}, (\pi_2 \pi_1)^{m+1} \mid (n, m) \in \mathbb{N}^2 \right\}$ *with* $\pi_1 = (a, \phi_1, b)$ *and* $\pi_2 = (b, \phi_2, a)$. $\Gamma_{\mathscr{A}_{loop}}^\omega$ *contains also* $(\pi_1 \pi_2)^\omega$ *and* $(\pi_2 \pi_1)^\omega$.

3 Abstract strategies

Several different definitions of the notion of strategy have been given in the literature. Here is a sampling.

- A strategy is a map F from terms to terms such that $t \mapsto F(t)$ ([6] in the context of the λ-calculus, [22] in the context of explicit substitutions calculi).

- A strategy is a sub ARS having the same set of normal forms ([7, 39] in the context of abstract reduction systems).

- A strategy is a plan for achieving a complex transformation using a set of rules ([43] in the context of program transformations).

- A strategy is a set of proof terms in rewriting logic ([11] in the ELAN system).

- A strategy is a (higher-order) function (in ELAN [10], Maude [37]) that can apply to other strategies.

- A strategy is a ρ-term in the ρ-calculus [15].

- A strategy is a subset of the set of all rewriting derivations [31], in the context of abstract strategies for deduction and computation. This view is further detailed below.

- A strategy is a partial function that associates to a reduction-in-progress, the possible next steps in the reduction sequence. Here, the strategy as a function depends only on the object and the derivation so far. This notion of strategy coincides with the definition of strategy in sequential path-building games, with applications to planning, verification and synthesis of concurrent systems [17].

Among these various definitions some of them are *extensional* in the sense that a strategy is taken explicitly as a set of derivations, while others are *intensional* in the sense that they describe a means for determining such a set. We focus in this paper on the two last definitions and explore their implications and their relations.

We use a general definition slightly different from the one used in [39]. This approach has already been proposed in [31] and here we essentially improve and detail a few definitions.

Definition 4 (Abstract Strategy) *Given an ARS $\mathscr{A}$, an* abstract strategy ζ *over $\mathscr{A}$ is a subset of non-empty derivations of $\Gamma_{\mathscr{A}}^{\omega}$.*

A strategy can be a finite or an infinite set of derivations, and the derivations themselves can be finite or infinite in length.

Let us introduce some terminology that will be useful later on.

Definition 5 (Factor-closed, prefix-closed, closed by composition) *Given an abstract strategy ζ over an ARS $\mathscr{A}$,*

- ζ *is* factor-closed *(resp.* prefix-closed*) iff for any derivation π in ζ, any factor (resp. prefix) of π is also in ζ.*

- ζ *is* closed by composition *iff for any two composable derivations π, π' in ζ, their composition $\pi\pi'$ is in ζ too.*

An abstract strategy over an abstract reduction system $\mathscr{A} = (\mathcal{O}, \mathscr{L}, \Gamma)$ induces a (partial) function from $\mathcal{O}$ to $2^{\mathcal{O}}$. This functional point of view has been already proposed in [11]; we just briefly recall it in our formalism.

The *domain* of a strategy ζ is the set of objects that are source of a derivation in ζ:

$$Dom(\zeta) = \bigcup_{\pi \in \zeta} Dom(\pi)$$

The application of a strategy is defined (only) on the objects of its domain. The application of a strategy ζ on $a \in Dom(\zeta)$ is denoted ζa and is defined as the set of all objects that can be reached from a using a finite derivation in ζ:

$$\zeta a = \{Im(\pi) \mid \pi \in \zeta, \pi \text{ finite and } Dom(\pi) = a\}$$

If $a \notin Dom(\zeta)$ we say that ζ *fails* on a (ζ contains no derivation of source a).

If $a \in Dom(\zeta)$ and $\zeta a = \varnothing$, we say that the strategy ζ is *indeterminate* on a. In fact, ζ is indeterminate on a if and only if $a \in Dom(\zeta)$ and ζ contains no finite derivation starting from a.

Example 3 (Strategies) *Let us consider the abstract reduction system $\mathscr{A}_{lc}$ of Example 1 and define the following strategies:*

1. *The strategy $\zeta_u = \Gamma_{\mathscr{A}_{lc}}^{\omega}$, also called the* Universal *strategy [31] (w.r.t. $\mathscr{A}_{lc}$), contains all the derivations of $\mathscr{A}_{lc}$. We have $\zeta_u a = \zeta_u b = \{a,b,c,d\}$ and ζ_u fails on c and d.*

2. *The strategy $\zeta_f = \varnothing$, also called* Fail*, contains no derivation and thus fails on any $x \in \{a,b,c,d\}$.*

3. *No matter which derivation of the strategy $\zeta_c = \left\{ \left(a \xrightarrow{\phi_1 \phi_3} a \right)^n a \xrightarrow{\phi_2} c \mid n \geq 0 \right\}$ is considered, the object a eventually reduces to c: $\zeta_c a = \{c\}$. ζ_c fails on b, c and d.*

4. *The strategy $\zeta_\omega = \left\{ \left(a \xrightarrow{\phi_1 \phi_3} a \right)^\omega \right\}$ is indeterminate on a and fails on b, c and d.*

The strategies ζ_u and ζ_f are prefix closed while ζ_c and ζ_ω are not.
The so-called Universal *and* Fail *strategies introduced in Example 3 can be obviously defined over any abstract reduction system.*

There is a natural topology on the space of derivations in an abstract reduction system. The set of (finite and infinite) derivations is essentially the *Kahn domain*[21] over the set of labels. The basic open sets in this topology are the "intervals", the sets $B_{\pi'} = \{\pi \mid \pi' \text{ is a prefix of } \pi\}$ as π' ranges over the finite derivations. Under this topology we have the following characterization of the *closed* sets.

Definition 6 *If ζ is a set of derivations, a* limit point *of ζ is a derivation π with the property that every finite prefix π_0 of π is a finite prefix of some derivation in ζ. A set ζ of (finite or infinite) derivations is* closed *if it contains all of its limit points. Equivalently: for every derivation π not in ζ, there is a finite prefix π_0 of π such that every extension of π_0 fails to be in ζ.*

As observed by Alpern and Schneider [1], the closed sets are precisely the *safety properties* [35] when derivations are viewed as runs of a system.

Example 4 *Let $\zeta = \left\{ \left(a \xrightarrow{\phi_1} b \xrightarrow{\phi_3} a \right)^n a \xrightarrow{\phi_2} c \mid n \in \mathbb{N} \right\}$ be an abstract strategy over $\mathscr{A}_{lc}$. The derivation $\left(a \xrightarrow{\phi_1} b \xrightarrow{\phi_3} a \right)^{\omega}$ is a limit point of ζ and does not belong to ζ. Thus, ζ is not closed.*

4 Intensional Strategies

In this section, strategies are considered as a way of constraining and guiding the steps of a reduction, possibly based on what has happened in the past. Under this reading, at any step in a derivation, we should be able to say whether a contemplated next step obeys the strategy ζ. This is in contrast to characterizing a set of reductions in terms of a global property, that may depend on an entire completed reduction. We introduce first strategies that do not take into account the past; although these memoryless strategies allow us to generate a significant number of classical abstract strategies used in term rewriting, they are less powerful when it comes to generate strategies ubiquitous in game theory. We introduce in what follows these two classes of strategies and will state formally in Section 5 their expressive power.

4.1 Memoryless strategies

Let us first consider in this section a class of strategies that chooses the next step only regarding the current object (or state). We follow an established convention [29] and call these strategies *memoryless*. The following definition formalizes the choice of the next step using a partial function on objects.

Definition 7 (Memoryless intensional strategy) *A memoryless intensional strategy over an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ is a partial function λ from $\mathscr{O}$ to 2^{Γ} such that for every object a, $\lambda(a) \subseteq \{\pi \mid \pi \in \Gamma, Dom(\pi) = a\}$.*

In this definition, $\pi \in \Gamma$ denotes a reduction step or equivalently a derivation of length 1.
A memoryless intensional strategy naturally generates an abstract strategy, as follows.

Definition 8 (Extension of a memoryless intensional strategy) *Let λ be a memoryless intensional strategy over an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$. The extension of λ is the abstract strategy ζ_{λ} consisting of the following set of derivations:*

$$\pi = ((a_i, \phi_i, a_{i+1}))_{i \in \mathfrak{I}} \in \zeta_{\lambda} \qquad iff \qquad \forall j \in \mathfrak{I}, \quad (a_j, \phi_j, a_{j+1}) \in \lambda(a_j)$$

We will sometimes say that the intensional strategy λ generates *the abstract strategy ζ_{λ}.*

This extension may obviously contain infinite derivations; in such a case it also contains all the finite derivations that are prefixes of the infinite ones. Indeed, it is easy to see from Definition 8 that the extension of an intensional strategy is closed under taking prefixes. We show next that the set of finite derivations generated by an intensional strategy λ can be constructed inductively from λ.

Definition 9 (Finite support of an abstract strategy) *Let us call the* finite support *of (any) strategy ζ the set of* finite *derivations in ζ and denote it $\zeta^{<\omega}$.*

Proposition 1 *Given a memoryless intensional strategy λ over an abstract reduction system of the form $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$, the finite support of its extension is an abstract strategy denoted $\zeta_\lambda^{<\omega}$ and inductively defined as follows:*

- $\bigcup_{a \in Dom(\lambda)} \lambda(a) \subseteq \zeta_\lambda^{<\omega}$
- $\forall \pi \in \zeta_\lambda^{<\omega}$ *and* $\forall \pi' \in \lambda(Im(\pi))$, $\pi\pi' \in \zeta_\lambda^{<\omega}$

Proof: Clearly, the derivations that are computed by this inductive definition are in ζ_λ and are finite. Conversely, by induction on the length of the derivations, any finite derivation in ζ_λ is built by one of the two inductive cases. $\square$

Notice that an intensional strategy λ over an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ induces a sub-ARS $\mathscr{B} = (\mathscr{O}, \mathscr{L}, \bigcup_{a \in Dom(\lambda)} \lambda(a))$ of $\mathscr{A}$ and thus, such that $\zeta_\lambda^{<\omega} = \Gamma_\mathscr{B}^+$ and $\zeta_\lambda = \Gamma_\mathscr{B}^\omega$. Note also that ζ_λ is the smallest closed strategy containing $\zeta_\lambda^{<\omega}$.

Example 5 *Let us consider the abstract reduction system $\mathscr{A}_{lc}$ of Example 1 and define the following strategies:*

- *The intensional strategy λ_u defined on all objects in $\mathscr{O}$ such that for any object $a \in \mathscr{O}$, $\lambda_u(a) = \{\pi \mid \pi \in \Gamma, Dom(\pi) = a\}$ obviously generates the Universal strategy ζ_u (of Example 3). Moreover $\zeta_{\lambda_u}^{<\omega} = \Gamma_{\mathscr{A}_{lc}}^+$.*

- *The intensional strategy λ_f defined on no object in $\mathscr{O}$ generates the Fail strategy ζ_f (of Example 3).*

- *Given an abstract reduction system $\mathscr{A}$, let us consider an order $<$ on the labels of $\mathscr{A}$ and a function "max" that computes the maximal element(s) of a set (the result is a singleton if the order is total). The intensional strategy λ_{gm} such that $\lambda_{gm}(a) = \{\pi : a \xrightarrow{\phi} b \mid \phi = max(\{\phi' \mid a \xrightarrow{\phi'} b \in \Gamma\})\}$ generates a "Greatmost" abstract strategy ζ_{gm} that, for each of its derivations, chooses each time one of the steps with the greatest "weight" specified by the label.*

  - *If we consider the abstract reduction system $\mathscr{A}_{lc}$ with the order $\phi_1 < \phi_2 < \phi_3 < \phi_4$, then*
    $$\zeta_{\lambda_{gm}} = \zeta_{\lambda_{gm}}^{<\omega} = \left\{ a \xrightarrow{\phi_2} c \; ; \; b \xrightarrow{\phi_4} d \right\}.$$
  - *If we consider the abstract reduction system $\mathscr{A}_{lc}$ with the order $\phi_1 > \phi_2 > \phi_3 > \phi_4$, then*
    $$\zeta_{\lambda_{gm}} = \left\{ \left(a \xrightarrow{\phi_1\phi_3} a \right)^n a \xrightarrow{\phi_1} b \; ; \; \left(a \xrightarrow{\phi_1\phi_3} a \right)^\omega \; ; \; \left(b \xrightarrow{\phi_3\phi_1} b \right)^n b \xrightarrow{\phi_3} a \; ; \; \left(b \xrightarrow{\phi_3\phi_1} b \right)^\omega \mid n \geq 0 \right\}$$
    and
    $$\zeta_{\lambda_{gm}}^{<\omega} = \left\{ \left(a \xrightarrow{\phi_1\phi_3} a \right)^n a \xrightarrow{\phi_1} b; \left(a \xrightarrow{\phi_1\phi_3} a \right)^m ; \left(b \xrightarrow{\phi_3\phi_1} b \right)^n b \xrightarrow{\phi_3} a; \left(b \xrightarrow{\phi_3\phi_1} b \right)^m \mid m > n \geq 0 \right\}$$

If the objects of the abstract reduction system are terms and the rewriting steps are labeled by the redex position, then we can use the prefix order on positions and the intensional strategy generates in this case the classical innermost *strategy. When a lexicographic order is used, the classical* rightmost-innermost *strategy is obtained.*

4.2 Intensional strategies with memory

The previous definition of memoryless intensional strategies cannot take into account the past derivation steps to decide the next possible ones. For that, the history of a derivation has to be memorized and available at each step. In order to define intensional strategies with memory, called simply intensional strategies in what follows, let us first introduce the notion of traced-object where each object memorizes how it has been reached.

Definition 10 (Traced-object) *Given a countable set of objects $\mathscr{O}$ and a countable set of labels $\mathscr{L}$ mutually disjoint, a* traced-object *is a pair $[\alpha]a$ where α is a sequence of elements of $\mathscr{O} \times \mathscr{L}$ called* trace *or* history *.*

In this definition, we implicitly define a monoid $\big((\mathscr{O} \times \mathscr{L})^*, \odot\big)$ generated by $(\mathscr{O} \times \mathscr{L})$ and whose neutral element is denoted by Λ.

Definition 11 (Traced object compatible with an ARS) *A traced-object $[\alpha]a$ is* compatible *with $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ iff $\alpha = ((a_i, \phi_i))_{i \in \mathfrak{I}}$ for any right-open interval $\mathfrak{I} \subseteq \mathbb{N}$ starting from 0 and $a = a_n$ and for all $i \in \mathfrak{I}$, $(a_i, \phi_i, a_{i+1}) \in \Gamma$. In such a case, we denote by $[\![\alpha]\!]$ the derivation $\big((a_i, \phi_i, a_{i+1})\big)_{i \in \mathfrak{I}}$ and by $\mathscr{O}^{[\mathscr{A}]}$ the set of traced objects compatible with $\mathscr{A}$. Moreover, we define an equivalence relation $\sim$ over $\mathscr{O}^{[\mathscr{A}]}$ as follows: $[\alpha]a \sim [\alpha']a'$ iff $a = a'$. We naturally have $\mathscr{O}^{[\mathscr{A}]}/\!\sim = \mathscr{O}$.*

We can now refine the definition of intensional strategies taking the history of objects into account.

Definition 12 (Intensional strategy (with memory)) *An intensional strategy over an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ is a partial function λ from $\mathscr{O}^{[\mathscr{A}]}$ to 2^Γ such that for every traced object $[\alpha]a$, $\lambda([\alpha]a) \subseteq \{\pi \in \Gamma \mid Dom(\pi) = a\}$.*

As for memoryless intensional strategies, an intensional strategy naturally generates an abstract strategy, as follows.

Definition 13 (Extension of an intensional strategy) *Let λ be an intensional strategy over an abstract reduction system $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$. The* extension *of λ is the abstract strategy ζ_λ consisting of the following set of derivations:*

$$\pi = ((a_i, \phi_i, a_{i+1}))_{i \in \mathfrak{I}} \in \zeta_\lambda \qquad \textit{iff} \qquad \forall j \in \mathfrak{I}, \quad (a_j, \phi_j, a_{j+1}) \in \lambda([\alpha]a_j)$$

where $\alpha = ((a_i, \phi_i))_{i \in \mathfrak{I}}$.

As before, we can inductively define the finite support of this extension as an abstract strategy $\zeta_\lambda^{<\omega}$ containing all finite derivations of ζ_λ.

Proposition 2 *Given an intensional strategy with memory λ over an abstract reduction system of the form $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$, the finite support of its extension is an abstract strategy denoted $\zeta_\lambda^{<\omega}$ inductively defined as follows:*

- *$\forall [\Lambda]a \in \mathscr{O}^{[\mathscr{A}]}, \lambda([\Lambda]a) \subseteq \zeta_\lambda^{<\omega}$,*

- *$\forall \alpha$ s.t. $\pi = [\![\alpha]\!] \in \zeta_\lambda^{<\omega}$ and $\pi' \in \lambda([\alpha]Im(\pi))$, $\pi\pi' \in \zeta_\lambda^{<\omega}$*

Proof: Similar as in Proposition 1. $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ $\square$

Example 6 *The following examples of strategies cannot be expressed without the knowledge of the history and illustrate the interest of traced objects.*

- *The intensional strategy that restricts the derivations to be of bounded length k makes use of the size of the trace α, denoted |α|:*

$$\lambda_{ltk}([\alpha]\,a) = \{\pi \mid \pi \in \Gamma, Dom(\pi) = a, |\alpha| < k-1\}$$

- *If we assume that the reduction steps are colored, for instance, in white or black via their labels, then the following intensional strategy generates a strategy whose reductions alternate white and black steps:*

$$\lambda_{WB}\big(\big[((a_i,\phi_i))_{0\leq i\leq n}\big]\,a\big) = \{\pi : a \xrightarrow{neg(\phi_n)} b \mid \pi \in \Gamma\}$$

with neg(white) = black and neg(black) = white. Once again, the knowledge of (the color) of the previous step is essential for choosing the current one.

- *The strategy that alternates reductions from a set (of steps) Γ_1 with reductions from a set Γ_2 can be generated by the following intensional strategy:*

$$
\begin{aligned}
\lambda_{\Gamma_1;\Gamma_2}([\Lambda]\,a) &= \{\pi_1 \mid \pi_1 \in \Gamma_1, Dom(\pi_1) = a\} \\
\lambda_{\Gamma_1;\Gamma_2}([\alpha' \odot (u,\phi')]\,a) &= \{\pi_1 \mid \pi_1 \in \Gamma_1, Dom(\pi_1) = a\} \quad \text{if } u \xrightarrow{\phi'} a \in \Gamma_2 \\
\lambda_{\Gamma_1;\Gamma_2}([\alpha' \odot (u,\phi')]\,a) &= \{\pi_2 \mid \pi_2 \in \Gamma_2, Dom(\pi_2) = a\} \quad \text{if } u \xrightarrow{\phi'} a \in \Gamma_1
\end{aligned}
$$

As a concrete example, let $\Gamma_1 = \{a \to b, b \to c\}$ and let $\Gamma_2 = \{b \to b\}$ (labels are omitted). Then, given the reduction $a \to b$, $\lambda_{\Gamma_1;\Gamma_2}$ yields $\{b \to b\}$ as next step, while given the reduction $a \to b \to b$, $\lambda_{\Gamma_1;\Gamma_2}$ yields $\{b \to c\}$ as next step. So $\lambda_{\Gamma_1;\Gamma_2}$ is not memoryless: if all we know is the last element of a history (for example, b) we cannot determine the next step(s).

- *Some standard term-rewriting strategies such as* parallel outermost *(when all outermost redexes must be contracted) or* Gross-Knuth reduction *are not memoryless when viewed as strategies over the reduction system whose steps are single-step rewrites.*

## 5   Expressiveness of intensional strategies

Not every abstract strategy arises as the extension of an intensional strategy. In this section we give a characterization of such abstract strategies.

5.1   Which abstract strategies can be described by intensional strategies ?

As a simple example, consider a single derivation π, and let ζ be the set of all prefixes of π (including π itself of course). Then ζ is the extension of the intensional strategy that maps each finite prefix of π to its next step, if there is one, and otherwise to the empty set. Similarly, any ζ consisting of the prefix-closures of a finite set of derivations is the extension of some intensional λ.

On the other hand, the following example is instructive.

Example 7 *Let $\mathscr{A}$ be the abstract reduction system consisting of two objects a and b and the steps $a \to a$ and $a \to b$ (the labels do not matter):*
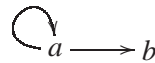
Let ζ be the set of all the reductions which eventually fire the rule $a \to b$. Then there is no intensional strategy λ such that $\zeta_\lambda = \zeta$. Suppose on the contrary that we had an intensional strategy determining

ζ. *Now ask: at a typical stage $a \to a \to \cdots \to a$ of a reduction, does the step of firing the rule $a \to a$ obey this strategy? Clearly the answer cannot be "no" at any stage since we would prevent ourselves from going further and firing the rule $a \to b$ eventually. On the other hand, if the answer is "yes" at every stage, then the infinite reduction $a \to a \to a \to a \to \ldots$ obeys our strategy at each step. But this reduction is not in ζ!*

Example 7 shows that not every set of reductions can be captured by an intensional notion of strategy. Note that this is not a result about computable strategies, or memoryless strategies, or deterministic strategies. And the set ζ is a perfectly reasonable set of derivations: it can be defined by the rational expression $(a \to a)^*(a \to b)$. But there is no *function on traced objects* generating precisely the set of derivations in question.

The intuition behind this example is this: for a given intensional strategy λ, if a derivation π *fails* to be in ζ_λ, then there is a particular step of π which fails to obey λ. This is the essential aspect of strategies that we suggest distinguishes them from other ways of defining sets of reductions: their local, finitary character.

As a preliminary to the following result, we show that the family of sets of reductions determined by strategies is closed under arbitrary intersection. Indeed, the following stronger observation is easy to verify.

Lemma 1 *Let $\Sigma = \{\lambda_i \mid i \in \mathfrak{I} \subseteq \mathbb{N}\}$ be any set of intensional strategies and λ the pointwise intersection of the λ_i, that is, $\lambda([\alpha]a) = \bigcap\{\lambda_i([\alpha]a) \mid i \in \mathfrak{I}\}$. Then $\zeta_\lambda = \bigcap\{\zeta_{\lambda_i} \mid i \in \mathfrak{I}\}$.*

Proof: An easy calculation based on definition 13. $\quad\square$

Proposition 3 *Let ζ be a set of (non-empty) derivations. There exists an intensional strategy λ with $\zeta_\lambda = \zeta$ iff ζ is a closed set.*

Proof: Let λ be an intensional strategy. To show that ζ_λ is closed, we must show that if π is a derivation that is not in ζ_λ, then there is a finite prefix π_0 of π such that every extension π_0' of π_0 fails to be in ζ_λ. Write π as the sequence of steps $s_1, s_2, \ldots$ where each s_i is an element $(a_i, \phi_i, a_{i+1}) \in \Gamma$. If π is not in ζ_λ, then for some i, the ith step $s_i = (a_i, \phi_i, a_{i+1}) \notin \zeta_\lambda([\alpha]a_i)$ where $\alpha = ((a_j, \phi_j))_{0 \le j < i}$. We can take π_0 to be $[\![\alpha]\!]$, the derivation composed of the $(i-1)$ first steps of π. Here we use the fact that if $\lambda([\alpha]a_i) = \varnothing$ then $\lambda([\alpha']a') = \varnothing$ for all α' such that $[\![\alpha']\!] = \pi' = \pi_0\pi_1$ for some π_1.

For the converse, suppose that ζ is a closed set of reductions. Then ζ is the intersection of the set of the complements of the basic open sets disjoint from ζ. By Lemma 1 it suffices to show that the complement of any basic open set is defined by an intensional strategy. So fix a finite π_0; we need to construct an intensional strategy determining the set of those π' which do *not* extend π_0. Letting $\pi_0 = s_1, s_2, \ldots, s_n$ we may simply define the strategy ζ to return the empty set on all π' extending π_0 and return all possible next moves on all other inputs. $\quad\square$

Example 8 *Consider the following abstract reduction system $\mathscr{A}$, modeling a simple intersection with two traffic signals. There are two directions (say, the north-south direction and the east-west direction). One traffic signal controls the north-south direction, another controls the east-west direction: each signal can be red or green. With each direction is associated a queue of cars waiting to cross: we can model these as natural numbers. (We can also bound the number of cars allowed in the model in order to obtain a finite abstract reduction system; this does not affect the observations made in this example.)*

So an object of $\mathscr{A}$ is a quadruple $[q_1, l_1, q_2, l_2]$ where q_1 and q_2 are natural numbers and l_1 and l_2 can each take the value 0 (for "red") or 1 (for "green").

The steps in $\mathscr{A}$ correspond to the fact that at each time unit

- *cars can arrive at queue 1 or queue 2*

- *some signal can change its value*

- *a car may cross the intersection if its signal is green*

This leads to the following set of labels: $\mathcal{L} = \{\mathsf{car}_1, \mathsf{car}_2, \mathsf{signal}_1, \mathsf{signal}_2, \mathsf{cross}_1, \mathsf{cross}_2\}$. *The steps of $\mathscr{A}$ may be defined by the following schematic reduction steps*

$$[q_1, l_1, q_2, l_2] \xrightarrow{\mathsf{car}_1} [(q_1 + 1), l_1, q_2, l_2] \qquad [q_1, l_1, q_2, l_2] \xrightarrow{\mathsf{car}_2} [q_1, l_1, (q_2 + 1), l_2]$$

$$[q_1, l_1, q_2, l_2] \xrightarrow{\mathsf{signal}_1} [q_1, (1 - l_1), q_2, l_2] \qquad [q_1, l_1, q_2, l_2] \xrightarrow{\mathsf{signal}_2} [q_1, l_1, q_2, (1 - l_2)]$$

$$[q_1, 1, q_2, l_2] \xrightarrow{\mathsf{cross}_1} [(q_1 - 1), 1, q_2, l_2] \qquad [q_1, l_1, q_2, 1] \xrightarrow{\mathsf{cross}_2} [q_1, l_1, (q_2 - 1), 1]$$

Note that $\mathscr{A}$ models the fact that cars may arrive at and cross the intersection in arbitrary patterns, and it reflects the constraint that the cars obey the traffic signals. But $\mathscr{A}$ does not, as an abstract reduction system, attempt to model an intelligent protocol for scheduling the traffic signals: it is an arena for developing and analyzing such a protocol.

The system admits some states that are intuitively undesirable, for example any state where both signals are green. There are also some derivations *that are undesirable, for example a derivation in which some car is left waiting at an intersection forever, such as the "unfair"*

$$[1, 0, 1, 1] \xrightarrow{\mathsf{cross}_2} [1, 0, 0, 1] \xrightarrow{\mathsf{car}_2} [1, 0, 1, 1] \xrightarrow{\mathsf{cross}_2} [1, 0, 0, 1] \ldots \tag{1}$$

An algorithm to manage the traffic signals is precisely an intensional strategy for the abstract reduction system $\mathscr{A}$, and the extension of such a strategy is the set of behaviors of the system that the strategy enforces.

Here are some results about strategies in this system; the second and fourth items are results that follow easily from the characterization theorem.

- *There are intensional strategies that ensure that the signals are never both green simultaneously (this is easy).*

- *More interestingly, there is an intensional strategy λ such that ζ_λ is precisely the set of those derivations such that the signals are never both green simultaneously. This is a highly non-deterministic strategy, that permits* any *behavior as long as it does not permit simultaneous greens.*

- *There are intensional strategies to ensure that any car arriving at an intersection is eventually allowed through.*

- *But there is* no *intensional strategy λ whose extension ζ_λ is precisely the set of all those paths in which any car arriving at an intersection is eventually allowed through.*

The second fact above follows from the observation that the set of derivations in which the signals are never both green simultaneously is a closed set (this is an easy consequence of Definition 6). The fourth fact above follows from the observation that the set of "fair" derivations, in which no car is forever denied access to the intersection, is not a closed set. To see this, note that the unfair derivation above is a limit point of the set of fair derivations, since every finite prefix of this derivation can be extended to one in which the car waiting (and all subsequent cars) crosses the intersection. Since the derivation itself is not fair, we see that the set of fair derivations does not contain all of its limit points, and so is not closed.

5.2 Closure properties for intensional strategies

In light of the importance of designing languages for expressing complex strategies, a natural question to ask is: what are the closure properties enjoyed by (the extensions of) intensional strategies? We observed in Lemma 1 that this class is closed under intersection. Indeed, by taking the pointwise intersection of a family $\{\lambda_i \mid i \in I\}$ of intensional strategies, the extensional strategy generated is the intersection of the extensions of the λ_i. However this pointwise construction fails for arbitrary union. Even when we restrict to finite unions the situation is subtle: if λ_1 and λ_2 are intensional strategies generating ζ_{λ_1} and ζ_{λ_2} respectively, and if we write $\lambda_1 \cup \lambda_2$ for the intensional strategy that is the pointwise union of λ_1 and λ_2, then $\lambda_1 \cup \lambda_2$ will *not,* in general, generate $\zeta_{\lambda_1} \cup \zeta_{\lambda_2}$, as the next example demonstrates.

Example 9 *Given the abstract reduction system with objects $\{a, b_1, b_2\}$ and reduction steps (a, ϕ_1, b_1), (a, ϕ_2, b_2), (b_1, β_1, a), (b_2, β_2, a), let λ_1 be the (memoryless) intensional strategy*

$$a \mapsto \{(a, \phi_1, b_1)\}, \quad b_1 \mapsto \{(b_1, \beta_1, a)\}$$

and let λ_2 be the intensional strategy

$$a \mapsto \{(a, \phi_2, b_2)\}, \quad b_2 \mapsto \{(b_2, \beta_2, a).$$

Clearly ζ_{λ_1} is the set of derivations that loop between a and b_1, and similarly for ζ_{λ_2}. If we now construct the intensional strategy $\lambda_1 \cup \lambda_2$ by taking the pointwise union of λ_1 and λ_2; thus

$$\lambda_1 \cup \lambda_2 \quad = \quad a \mapsto \{(a, \phi_1, b_1), (a, \phi_2, b_2)\}, b_1 \mapsto \{(b_1, \beta_1, a)\}, b_2 \mapsto \{(b_2, \beta_2, a)$$

then clearly $\zeta_{\lambda_1 \cup \lambda_2}$ is the set of all derivations. Thus

$$\zeta_{\lambda_1 \cup \lambda_2} \neq \zeta_{\lambda_1} \cup \zeta_{\lambda_2}$$

That is, the pointwise union of intensional strategies does not give rise to the union of the corresponding extensions.

Nevertheless, $\zeta_{\lambda_1} \cup \zeta_{\lambda_2}$ is indeed generated by an intensional strategy (with memory): at the first step at object a we non-deterministically move to b_1 or to b_2 and at subsequent steps we always make the same choice. It is not an accident that we can generate $\zeta_{\lambda_1} \cup \zeta_{\lambda_2}$ intensionally, as we see next.

Proposition 4 *The class of abstract strategies that are extensions of intensional strategies is closed under arbitrary intersection and finite unions; it is not closed under complement.*

Proof: These are immediate consequences of Proposition 3 and basic facts about topological spaces. □

The fact that unions of extensions of intensional strategies are intensionally generated–even though the naive "pointwise union" construction fails– is a nice application of the topological perspective.

5.3 Other classes of abstract strategies that can be intensionally described

When restricting to abstract strategies consisting of a potentially infinite number of finite derivations, the existence of a corresponding intensional strategy depends on some classical properties of the original strategy. The proof of existence of a corresponding intensional strategy under appropriate assumptions explicits the way such a strategy can be built.

Proposition 5 *Given an abstract strategy ζ over $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ consisting only of finite derivations, there is a memoryless intensional strategy λ over $\mathscr{A}$ such that $\zeta_{\lambda}^{<\omega} = \zeta$ iff ζ is factor-closed and closed under composition.*

Proof: Obviously, the abstract strategy $\zeta_{\lambda}^{<\omega}$ built from an intensional strategy λ is factor-closed and closed by composition. Conversely, if ζ is factor-closed and closed by composition, λ can be defined as follows: for all object a, $\lambda(a) = \{a \xrightarrow{\phi} b \mid a \xrightarrow{\phi} b \in \zeta \cap \Gamma.\}$ $\square$

The existence of memoryless intensional strategy obviously implies the existence of an intensional strategy (with memory). The class of abstract strategies that satisfy the above conditions is already quite important, especially when considering term rewriting strategies but, as we have seen in Section 4, there are strategies that do not fit these constraints.

For example, if we consider again Example 8 over a finite interval of time, we can find a memoryless intensional strategy that corresponds to the set of derivations such that the signals are never both green simultaneously. On the other hand there is an intensional strategy, but not a memoryless one, for the set of derivations such that a car waits at most n turns before crossing the intersection. This follows from the following proposition.

Proposition 6 *Given an abstract strategy ζ over $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ consisting only of finite derivations, there is an intensional strategy λ over $\mathscr{A}$ such that $\zeta_{\lambda}^{<\omega} = \zeta$ iff ζ is prefix-closed.*

Proof: Let λ be an intensional strategy over $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$ and $\pi \in \zeta_{\lambda}^{<\omega}$; two cases are possible:

- $|\pi| \leq 1$, then π has no prefix.
- $|\pi| > 1$, then there exists α s.t. $\pi_{pref} = [\![\alpha]\!] \in \zeta_{\lambda}^{<\omega}$ and $\pi' \in \lambda([\alpha] Im(\pi))$, $\pi = \pi_{pref}\pi'$. By applying the same reasoning over π_{pref}, we obtain that all prefixes of π are in $\zeta_{\lambda}^{<\omega}$.

Conversely, let us consider a prefix-closed abstract strategy ζ. The intensional strategy λ defined as follows:

- $\forall a \in \mathscr{O}, \lambda([\Lambda] a) = \{\pi \in \zeta \cap \Gamma \mid Dom(\pi) = a\}$,
- $\forall \pi \in \zeta$, and $[\alpha] a \in \mathscr{O}^{[\mathscr{A}]}$ s.t. $[\![\alpha]\!] = \pi$, $\lambda([\alpha] a) = \{\pi' \in \Gamma \mid \pi\pi' \in \zeta\}$

is such that $\zeta_{\lambda}^{<\omega} = \zeta$. $\square$

6   Logical intensional strategies

Instead of defining an intensional strategy λ by a function, we can consider using a logical approach and associating to λ a *characteristic property* denoted by $\mathscr{P}_{\lambda}$ such that:

$$\mathscr{P}_{\lambda}([\alpha] a, \phi) \text{ is true iff } \exists b \text{ such that } (a, \phi, b) \in \lambda([\alpha] a)$$

Thus, $\zeta_{\lambda}^{<\omega}$ is the following prefix closed abstract strategy:

$$\left\{ [\![\alpha \odot (a, \phi)]\!] \mid \mathscr{P}_{\lambda}([\alpha] a, \phi) \text{ and } (\alpha = \Lambda \vee [\![\alpha]\!] \in \zeta_{\lambda}^{<\omega}) \text{ and } \exists b, a \xrightarrow{\phi} b \in \Gamma \right\}$$

Example 10 *Let us show how some of the previous examples and some variants can be easily expressed with a characteristic property.*

- *For the Universal strategy λ_u over an ARS $\mathscr{A} = (\mathscr{O}, \mathscr{L}, \Gamma)$, $\mathscr{P}_u([\alpha] a, \phi) = \top$, where $\top$ denotes the true Boolean value.*

- *For the Fail strategy λ_f over an ARS $\mathscr{A} = (\mathcal{O}, \mathscr{L}, \Gamma)$, $\mathscr{P}_f([\alpha] a, \phi) = \bot$, where $\bot$ denotes the false Boolean value.*

- *For the Greatmost strategy λ_{gm} over an ARS $\mathscr{A} = (\mathcal{O}, \mathscr{L}, \Gamma)$ with an order $<$ on the labels,*
  $$\mathscr{P}_{gm}([\alpha] a, \phi) = \forall (\phi', b), a \xrightarrow{\phi'} b \in \Gamma \Rightarrow \phi \not< \phi'.$$

- *For the strategy λ_{ltk} that selects the set of derivations of length at most k, $\mathscr{P}_{ltk}([\alpha] a, \phi) = |\alpha| < k$.*

- *For the strategy $\lambda_{R_1;R_2}$ that alternates reductions with labels from $R_1 \in \mathscr{L}$ with reductions with labels from $R_2 \in \mathscr{L}$,*

$$\mathscr{P}_{R_1;R_2}([\alpha] a, \phi) = \left\{ \begin{array}{lll} & \alpha = \Lambda \Rightarrow & (\phi \in R_1 \vee \phi \in R_2) \\ \wedge & \alpha = \alpha' \odot (u, \phi') \Rightarrow & ((\phi' \in R_1 \Rightarrow \phi \in R_2) \vee (\phi' \in R_2 \Rightarrow \phi \in R_1)) \end{array} \right\}$$

Indeed, using a logical property instead of a fonction is rather a matter of choice or can be related to the properties of strategies we want to study, but this does not bring more expressivity. As previously said, intensional strategies generate only closed sets of derivations and thus always contain all prefixes of derivations. This prevents us from computing extensional strategies that look straightforward like the one in the next example.

Example 11 *We consider again the abstract reduction system $\mathscr{A}_{lc}$ and a strategy reduced to only one derivation $\zeta = \{a \xrightarrow{\phi_1} b \xrightarrow{\phi_3} a \xrightarrow{\phi_2} c\}$. ζ cannot be computed by an intensional strategy λ built as before since its extension would contain too many derivations, namely all prefixes of the derivation in ζ.*

In order to avoid this constraint, we characterize *accepted derivations* belonging to ζ_λ by defining a property over $\mathcal{O}^{[\mathscr{A}]}$ called *accepting states* and denoted by $\mathscr{F}_\lambda$. This leads to the following extended definition of an intensional strategy:

Definition 14 *We call* logical intensional strategy *over $\mathscr{A} = (\mathcal{O}, \mathscr{L}, \Gamma)$ any pair $(\lambda, \mathscr{F}_\lambda)$ where λ is an intensional strategy with memory and $\mathscr{F}_\lambda \subseteq \mathcal{O}^{[\mathscr{A}]}$.*

A logical intensional strategy $(\lambda, \mathscr{F}_\lambda)$ generates the abstract strategy $\{[\![\alpha]\!] \in \zeta_\lambda \mid [\alpha] Im(\alpha) \in \mathscr{F}_\lambda\}$.

In practice, it may be useful to describe the set $\mathscr{F}_\lambda$ by its characteristic function F_λ and to test whether $F_\lambda([\alpha] Im(\alpha)) = true$.

Let us illustrate on simple examples the expressive power gained in this extended definition.

Example 12 *Coming back to Example 11, we can now characterize the only derivation of interest by simply stating that $\mathscr{F}_\lambda$ contains only the compatible traced-object $[(a, \phi_1) \odot (b, \phi_3) \odot (a, \phi_2)] c$.*

In order to define a strategy that selects derivations of length greater than k, we cannot proceed as for defining λ_{ltk} since the strategy is not prefix closed, but we can characterize accepting states, namely those reached in more than k steps: $\mathscr{F}_{gtk} = \{[\alpha] a \mid |\alpha| \geq k\}$. The situation is similar when one wants to define a strategy that selects derivations of length exactly k.

As a last example, we can now formalize the strategy of Example 7 with the following accepting states $\mathscr{F}_\lambda = \{[\alpha] b \mid \exists n \in \mathbb{N}, [\![\alpha]\!] : a \xrightarrow{n} a \rightarrow b\}$.

7 Conclusion

We have proposed and discussed in this paper different definitions of strategies stressing different aspects: clearly the notion of abstract strategy is appropriate to explore semantic properties, while intensional

strategies are more adequate for operational purposes. We have tried to show how these two views do not exclude each other but rather may be very complementary.

In order to express interesting strategies in an operational way, we have introduced traced objects that memorize their history. There is an interesting analogy with languages which is worth exploring: derivations (histories) are words built on 3-tuples (a, ϕ, b); the strategy (seen as a set of derivations) is the language (set of words) to recognize; the characteristic property of the strategy is the way to decide whether a word belongs to the language. Based on this analogy, it would be interesting to see if it is possible to characterize classes of recognizable and computable strategies.

Another direction we want to explore is the definition of intensional strategies that, at a given step, can look forward in the following intended derivation steps. Formalizing such looking-forward intensional strategies is motivated for instance by looking-ahead mechanisms in constraint solving.

Finally, we have only sketched the definition of intensional strategies with accepting conditions and this approach needs further work.

The domain of strategic reductions is yet largely unexplored and important questions remain. Let us mention further topics that have not been addressed in this paper but we think interesting to explore.

- An important topic that requires yet further exploration is the design of a *strategy language* whose purpose is to give syntactic means to describe strategies. Actually, several systems based on rewriting already provide a strategy language, for instance ELAN [32, 11], Stratego [42], Strafunski [34], TOM [5] or more recently Maude [37]. It is interesting to identify common constructs provided in these different languages, and to try to classify them according to their use either to explore the structure of objects (here terms) or to build derivations or sets of derivations.

  - Basic constructions are given by rewrite rules whose application corresponds to an elementary reduction step. Identity and failure are also present as elementary constructions. Their semantics is given by their abstract strategy definitions.

  - Due to the tree structure of terms, traversal strategies that give access to sub-terms are based on two constructions *All* and *One* that consider immediate sub-terms of a given node: on a term t, $All(s)$ applies the strategy s on all immediate sub-terms, while $One(s)$ applies the strategy s on the first immediate sub-term where s does not fail.

  - Operations to build derivations are sequential composition *Sequence* and choice, that may be deterministic (*Choice*) or not (*ND − Choice*). Another construction, *Try*, which gives an unfailing choice, is very useful and can be just derived from the previous ones.

  - With a functional view of strategies, it is natural to define recursive strategies and to define a fixpoint operator. This is the way to perform iteration and for example to construct the *Repeat* operator.

  - Traversal strategies also use the fixpoint operator to program different ways to go through the term structure, as in *Innermost* or *Outermost* strategies. Such traversals are typical intensional strategies, as described in this paper.

  These constructions are devoted to rewriting on terms or term graphs. Indeed, extending the language to graph rewriting raises new challenges, such as graph traversal. Other constructions could be interesting in more general contexts than term rewriting, especially parallel application of strategies with indeed non-interference.

- Proving properties of strategies and strategic reductions has already been explored in the case of specific strategies. Let us mention in particular the following approaches for specific properties: confluence, weak and strong termination, completeness of strategic rewriting have been addressed

in [28, 27, 26, 23] for several traversal strategies using a schematization of derivation trees and an inductive proof argument. Another approach is the dependency pairs technique which has been adapted to prove termination of rewriting under innermost[3] or lazy strategies[25]. Other works such as [24, 41] have considered strategies transformation to equivalent rewrite systems that are preserving properties like termination. However other properties, such as fairness or loop-freeness, have been much less studied. In general, we may expect that the logical characterization of intensional strategies could help to prove derivation properties in the context of strategic derivations.

- We have distinguished between arbitrary intensional strategies and *memoryless* strategies. In the well-studied domain of games on finite graphs–so important in verification research—there is an important middle ground between these: the class of strategies requiring a fixed finite amount of memory. Such strategies can be computed by a finite-state machine [19]. The question of finite-memory strategies over abstract reductions systems is subtle, essentially due to the fact that an abstract reduction system corresponds to a (solitaire) game on a typically *infinite* arena. Note that even a conceptually simple term-rewriting strategy such as parallel-outermost cannot be said to require a fixed finite amount of memory, because terms can have an unbounded number of parallel-outermost redexes (see Example 6). A careful treatment of the proper analogue of *finite-memory* strategies over general abstract reduction systems is an interesting topic for future work.

References

[1] Bowen Alpern & Fred B. Schneider (1987): *Recognizing Safety and Liveness*. *Distributed Computing* 2(3), pp. 117–126.

[2] André Arnold, Aymeric Vincent & Igor Walukiewicz (2003): *Games for synthesis of controllers with partial observation*. *Theoretical Computer Science* 303(1), pp. 7–34.

[3] Thomas Arts & Jürgen Giesl (2000): *Termination of term rewriting using dependency pairs*. *Theoretical Computer Science* 236, pp. 133–178.

[4] Franz Baader & Tobias Nipkow (1998): *Term Rewriting and all That*. Cambridge University Press.

[5] Emilie Balland, Paul Brauner, Radu Kopetz, Pierre-Etienne Moreau & Antoine Reilles (2006): Tom Manual. LORIA, Nancy (France), version 2.4 edition.

[6] Henk Barendregt (1984): *The Lambda-Calculus, its syntax and semantics*. Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam. Second edition.

[7] E. Barendsen, I. Bethke, J. Heering, R. Kennaway, P. Klint, V. van Oostrom, F. van Raamsdonk, F-J. de Vries & H. Zantema (2003): *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

[8] Dietmar Berwanger, Erich Grädel & Stephan Kreutzer (2003): *Once upon a Time in the West - Determinacy, Definability, and Complexity of Path Games*. In: Moshe Y. Vardi & Andrei Voronkov, editors: *LPAR*, *Lecture Notes in Computer Science* 2850. Springer, pp. 229–243.

[9] Andreas Blass (1992): *A Game Semantics for Linear Logic*. *Annals of Pure and Applied Logic* 56(1-3), pp. 183–220.

[10] Peter Borovanský, Claude Kirchner, Hélène Kirchner & Pierre-Etienne Moreau (2002): *ELAN from a rewriting logic point of view*. *Theoretical Computer Science* 2(285), pp. 155–185.

[11] Peter Borovanský, Claude Kirchner, Hélène Kirchner & Christophe Ringeissen (2001): *Rewriting with strategies in* ELAN*: a functional semantics*. *International Journal of Foundations of Computer Science* 12(1), pp. 69–98.

[12] Paul Brauner, Clément Houtmann & Claude Kirchner (2007): *Principle of superdeduction*. In: Luke Ong, editor: *22nd IEEE Symposium on Logic in Computer Science (LICS 2007)*. pp. 41–50.

[13] Paul Brauner, Clément Houtmann & Claude Kirchner (2007): *Superdeduction at work*. In: H. Comon-Lundh, C. Kirchner & H. Kirchner, editors: *Rewriting, Computation and Proof. Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science 4600. Springer, pp. 132–166.

[14] Carlos Castro (1998): *Building Constraint Satisfaction Problem Solvers Using Rewrite Rules and Strategies*. Fundamenta Informaticae 34, pp. 263–293.

[15] Horatiu Cirstea & Claude Kirchner (2001): *The rewriting calculus — Part I and II*. Logic Journal of the Interest Group in Pure and Applied Logics 9(3), pp. 427–498.

[16] David Delahaye (2000): *A tactic language for the system Coq*. In: Michel Parigot & Andrei Voronkov, editors: *Proceedings of Logic Programming and Automated Reasoning (LPAR)*, Lecture Notes in Computer Science 1955. Springer, pp. 85–95.

[17] Daniel J. Dougherty (2008): *Rewriting strategies and game strategies*. Internal report.

[18] Gilles Dowek, Thérèse Hardin & Claude Kirchner (2003): *Theorem Proving Modulo*. Journal of Automated Reasoning 31(1), pp. 33–72.

[19] Stefan Dziembowski, Marcin Jurdzinski & Igor Walukiewicz (1997): *How Much Memory is Needed to Win Infinite Games?* In: *12th IEEE Symposium on Logic in Computer Science (LICS 1997)*. pp. 99–110.

[20] Steven Eker, Narciso Martí-Oliet, José Meseguer & Alberto Verdejo (2007): *Deduction, Strategies and Rewriting*. Electronics Notes in Theoretical Computer Science 174(11), pp. 3–25.

[21] Martín Hötzel Escardó (2004): *Synthetic Topology: of Data Types and Classical Spaces*. In: *Proceedings of the Workshop on Domain Theoretic Methods for Probabilistic Processes*, Electronics Notes in Theoretical Computer Science 87. Elsevier Science Publishers B. V. (North-Holland), pp. 21–156.

[22] Maribel Fernández, Ian Mackie & François-Régis Sinot (2005): *Closed reduction: explicit substitutions without alpha-conversion*. Mathematical Structures in Computer Science 15(2), pp. 343–381.

[23] Olivier Fissore, Isabelle Gnaedig & Hélène Kirchner (2002): *Outermost Ground Termination*. In: *Proceedings of the 4th International Workshop on Rewriting Logic and Its Applications*, Electronic Notes in Theoretical Computer Science 71. Elsevier Science Publishers B. V. (North-Holland), Pisa, Italy, pp. 188–207.

[24] Olivier Fissore, Isabelle Gnaedig & Hélène Kirchner (2003): *Simplification and Termination of Strategies in Rule-Based Languages*. In: *Proceedings of the Fifth International Conference on Principles and Practice of Declarative Programming*. ACM Press, Uppsala (Sweden), pp. 124–135.

[25] Jürgen Giesl, Stephan Swiderski, Peter Schneider-Kamp & René Thiemann (2006): *Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages*. In: *Proceedings of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science 4098. Springer, pp. 297–312.

[26] Isabelle Gnaedig & Hélène Kirchner (2006): *Computing Constructor Forms with Non Terminating Rewrite Programs*. In: *Proceedings of the Eighth ACM-SIGPLAN International Symposium on Principles and Practice of Declarative Programming*. ACM Press, Venice, Italy, pp. 121–132.

[27] Isabelle Gnaedig & Hélène Kirchner (2007): *Narrowing, Abstraction and Constraints for Proving Properties of Reduction Relations*. In: H. Comon-Lundh, C. Kirchner & H. Kirchner, editors: *Rewriting, Computation and Proof. Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science 4600. Springer, pp. 44–67.

[28] Isabelle Gnaedig & Hélène Kirchner (2009): *Termination of rewriting under strategies*. ACM Trans. Comput. Logic 10(2), pp. 1–52.

[29] Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001]*, Lecture Notes in Computer Science 2500. Springer.

[30] Alexander S. Kechris (1995): *Classical Descriptive Set Theory*, Graduate Texts in Mathematics 156. Springer.

[31] Claude Kirchner, Florent Kirchner & Hélène Kirchner (2008): *Strategic Computations and Deductions*. In:

Christoph Benzmüller, Chad E. Brown, Jörg Siekmann & Richard Statman, editors: *Reasoning in Simple Type Theory. Festchrift in Honour of Peter B. Andrews on His 70th Birthday*, *Studies in Logic and the Foundations of Mathematics* 17. College Publications, pp. 339–364.

[32] Claude Kirchner, Hélène Kirchner & Marian Vittek (1995): *Designing Constraint Logic Programming Languages using Computational Systems*. In: P. Van Hentenryck & V. Saraswat, editors: *Principles and Practice of Constraint Programming. The Newport Papers.*, chapter 8. The MIT Press, pp. 131–158.

[33] Jan Willem Klop, Vincent van Oostrom & Femke van Raamsdonk (1993): *Combinatory reduction systems: introduction and survey*. *Theoretical Computer Science* 121, pp. 279–308.

[34] Ralf Lämmel & Joost Visser (2002): *Design Patterns for Functional Strategic Programming*. In: *Proc. of Third ACM SIGPLAN Workshop on Rule-Based Programming RULE'02*. ACM Press, Pittsburgh, USA.

[35] Leslie Lamport (1977): *Proving the correctness of multiprocess programs*. *IEEE Transactions on Software Engineering* 3(2), pp. 125–143.

[36] Jean-Jacques Lévy (1978): *Réductions correctes et optimales dans le lambda-calcul*. Ph.D. thesis, Université de Paris VII.

[37] Narciso Martí-Oliet, José Meseguer & Alberto Verdejo (2005): *Towards a Strategy Language for Maude*. In: Narciso Martí-Oliet, editor: *Proceedings Fifth International Workshop on Rewriting Logic and its Applications, WRLA 2004, Barcelona, Spain, March 27 – April 4, 2004*, *Electronic Notes in Theoretical Computer Science* 117. Elsevier Science Publishers B. V. (North-Holland), pp. 417–441.

[38] Vincent van Oostrom & Roel de Vrijer (2003): *Term Rewriting Systems*, 2, chapter 8: Equivalence of reductions. Cambridge University Press.

[39] Vincent van Oostrom & Roel de Vrijer (2003): *Term Rewriting Systems*, 2, chapter 9: Strategies. Cambridge University Press.

[40] Marco Pistore & Moshe Y. Vardi (2003): *The Planning Spectrum - One, Two, Three, Infinity*. In: *18th IEEE Symposium on Logic in Computer Science (LICS 2003)*. IEEE Computer Society, pp. 234–243.

[41] Matthias Raffelsieper & Hans Zantema (2009): *A Transformational Approach to Prove Outermost Termination Automatically*. In: *Proceedings of the 8th International Workshop on Reduction Strategies in Rewriting and Programming*, *Electronic Notes in Theoretical Computer Science* 237. Elsevier Science Publishers B. V., pp. 3–21.

[42] Eelco Visser (2001): *Stratego: A Language for Program Transformation based on Rewriting Strategies. System Description of Stratego 0.5*. In: A. Middeldorp, editor: *Rewriting Techniques and Applications (RTA'01)*, *Lecture Notes in Computer Science* 2051. Springer, pp. 357–361.

[43] Eelco Visser (2005): *A survey of strategies in rule-based program transformation systems*. *Journal of Symbolic Computation* 40(1), pp. 831–873.

[44] Igor Walukiewicz (2004): *A Landscape with Games in the Background*. In: *19th IEEE Symposium on Logic in Computer Science (LICS 2004)*. IEEE Computer Society, pp. 356–366.