ISR 2014
Strategies

Hélène KIRCHNER
Inria

August 2014

Strategy Semantics: Rewriting logic

Objectives and Contents

Objectives of this lecture :
Survey different definitions of strategies in the rewriting context, show their relations. Illustrate the progression of ideas.

Contents :

1. Rewriting logic
2. Rewriting calculus
3. Abstract reduction systems

Rewriting Logic
[due to J. Meseguer TCS92]

Rewriting logic (RL) is a natural model of computation and an expressive semantic framework for concurrency, parallelism, communication, and interaction. It can be used for specifying a wide range of systems and languages in various application fields. It also has good properties as a metalogical framework for representing logics. In recent years, several languages based on RL (ASF+SDF, CafeOBJ, ELAN, Maude) have been designed and implemented.

Page of WRLA 2012 `http://wrla2012.lcc.uma.es/`

Formulas are sequents of the form

$$\pi : t \to t'$$

where π is a proof term, built on $\mathcal{F} \cup \mathcal{L} \cup \{;\}$
recording the proof of the sequent.

$$\mathcal{R} \vdash \pi : t \to t'$$

if $\pi : t \to t'$ can be obtained by finite application of the following deduction rules

Reflexivity For any $t \in \mathcal{T}(\mathcal{F})$:

$$t : t \to t$$

Congruence For any $f \in \mathcal{F}$ with $arity(f) = n$:

$$\frac{\pi_1 : t_1 \to t'_1 \quad \dots \quad \pi_n : t_n \to t'_n}{f(\pi_1, \dots, \pi_n) : f(t_1, \dots, t_n) \to f(t'_1, \dots, t'_n)}$$

Transitivity

$$\frac{\pi_1 : t_1 \to t_2 \qquad \pi_2 : t_2 \to t_3}{\pi_1 ; \pi_2 \ : \ t_1 \to t_3}$$

Replacement For any $\ell(x_1, \dots, x_n) : l \Rightarrow r \in \mathcal{R}$,

$$\frac{\pi_1 : t_1 \to t'_1 \quad \dots \quad \pi_n : t_n \to t'_n}{\ell(\pi_1, \dots, \pi_n) : l(t_1, \dots, t_n) \to r(t'_1, \dots, t'_n)}$$

A proof term example

```
rules for List
X, Y : Nat ; L L' L'' : List;
rec : sort (L X L' Y L'') => sort (L Y L' X L'')
if Y < X
fin : sort (L) => L
end
```

```
sort (3 1 2) -> sort (1 3 2) -> sort (1 2 3) -> (1 2 3)
```

```
rec(nil,3,nil,1,(2));rec((1),3,nil,2,nil);fin((1 2 3))
```

Strategies in rewriting logic

- A proof term π encodes a derivation
- A strategy is a set of proof terms in rewriting logic (Elan).

$$S = \{\pi | \pi \in \mathcal{PT}\}$$

- A strategy is a higher-order function : applying the strategy S to the term t means finding all terms t' such that

$$\pi : t \to t' \mid \pi \in S$$

- Since rewriting logic is reflective, strategy semantics can be defined inside the logic by rewrite rules at the meta-level (Maude approach).

ISR 2014
Strategies

Hélène KIRCHNER
Inria

August 2014

Strategy Semantics: Rewriting calculus

Rewriting Calculus

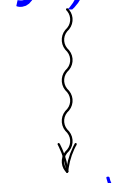
Introduced in 1998 by Horatiu Cirstea and Claude Kirchner

The rho-calculus has been introduced as a general means to uniformly integrate rewriting and lambda calculus. This calculus makes explicit and first-class all of its components: matching (possibly modulo given theories), abstraction, application and substitutions.
The rho-calculus is designed and used for logical and semantical purposes. It could be used with powerful type systems and for expressing the semantics of rule based as well as object oriented paradigms. It allows one to naturally express exceptions and imperative features as well as expressing elaborated rewriting strategies.

Page `http://rho.loria.fr/index.html`

Term rewriting

$$f(x,y) \Rightarrow x$$



$$f(a,b) \longrightarrow a$$

$$f(x,y) \Rightarrow x \bullet f(a,b) \longrightarrow a$$

Term rewriting - Exercise

```
rules for List
X, Y : Nat ; L L' L'' : List;
rec  : sort (L X L' Y L'') => sort (L Y L' X L'')
if Y < X
fin  : sort (L) => L
end
```

```
sort (3 1 2) -> sort (1 3 2) -> sort (1 2 3) -> (1 2 3)
```

Exercise : write the proof term in ρ-calculus

$$sort(L_3) \Rightarrow L_3 \bullet sort(L_2 X_2 L_2' Y_2 L_2'') \Rightarrow sort(L_2 Y_2 L_2' X_2 L_2'') \bullet$$
$$sort(L_1 X_1 L_1' Y L_1'') \Rightarrow sort(L_1 Y_1 L_1' X_1 L_1'') \bullet sort(312)$$

Term rewriting - Exercise

```
rules for List
X, Y : Nat ; L L′ L″ : List;
rec : sort (L X L′ Y L″) => sort (L Y L′ X L″)
if Y < X
fin : sort (L) => L
end
```

```
sort (3 1 2) -> sort (1 3 2) -> sort (1 2 3) -> (1 2 3)
```

Exercise : write the proof term in ρ-calculus

$sort(L_3) \Rightarrow L_3 \bullet sort(L_2 X_2 L_2' Y_2 L_2'') \Rightarrow sort(L_2 Y_2 L_2' X_2 L_2'') \bullet$
$sort(L_1 X_1 L_1' Y L_1'') \Rightarrow sort(L_1 Y_1 L_1' X_1 L_1'') \bullet sort(312)$

A calculus with more explicit features

In “basic” rewriting calculus,

- rules are first class object
- application is explicit
- decision of redex reduction is explicit
- matching is a main explicit parameter
- results are first class objects

Strategies in ρ-calculus

- Terms and λ-terms are ρ-terms ($\lambda x.t$ is $(x \Rightarrow t)$)
- A rule is a ρ-term
- A strategy is a ρ-term
- Rules and strategies are abstractions of the same nature!
- Application generalizes $\beta-$reduction
- Composition of strategies like function composition
- Recursion with μ operator.

$$\mu x.s = s[x \leftarrow \mu x.s]$$

The Abstract Biochemical Calculus

Initiated in Oana Andrei's Phd in 2007

A rewriting calculus that models an autonomous system as a *biochemical program* :

- collections of molecules (objects and rewrite rules)
- higher-order rewrite rules over molecules (that may introduce new rewrite rules in the system)
- strategies for modelling the system's evolution

A visual representation via *port graphs* and an implementation is provided by the PORGY environment.

Strategies are Abstract molecules

Definition of useful strategies:

$$
\begin{array}{rcl}
\mathsf{id} & \triangleq & X \Rightarrow X \\
\mathtt{fail} & \triangleq & X \Rightarrow \mathsf{stk} \\
\mathtt{seq}(S_1, S_2) & \triangleq & X \Rightarrow S_2 \bullet (S_1 \bullet X) \\
\mathtt{first}(S_1, S_2) & \triangleq & X \Rightarrow (S_1 \bullet X)\ \ (\mathsf{stk} \Rightarrow (S_2 \bullet X)) \bullet (S_1 \bullet X) \\
\mathtt{try}(S) & \triangleq & \mathtt{first}(S, \mathsf{id}) \\
\mathtt{not}(S) & \triangleq & X \Rightarrow \mathtt{first}(\mathsf{stk} \Rightarrow X, X' \Rightarrow \mathsf{stk}) \bullet (S \bullet X) \\
\mathtt{ifTE}(S_1, S_2, S_3) & \triangleq & X \Rightarrow \mathtt{first}(\mathsf{stk} \Rightarrow S_3 \bullet X, X' \Rightarrow S_2 \bullet X) \bullet (S_1 \bullet X) \\
\mathtt{repeat}(S) & \triangleq & \mu X.\mathtt{try}(\mathtt{seq}(S, X))
\end{array}
$$

Allows failure handling, repair instructions, persistent application,...
and more generally strategies for autonomic computing.

Strategies for autonomic computing

In autonomic computing, systems and their components have to:

- (re-)configure themselves automatically according to directives (rewrite rules and strategies): *self-configuration*
- must be prepared to face functioning problems and malicious attacks or failure: *self-protection*
- must repair themselves: *self-healing*
- seek new ways of optimizing their performance and efficiency *via* new rewrite rules and strategies that they deduce: *self-optimization*

Application : embedding invariant verification

An invariant property is encoded as a special rule in the biochemical program modelling the system and such rule is dynamically checked at each execution step.

- An invariant of the system : $G \Rightarrow G$.
  The strategy verifying such an invariant:
  $$\texttt{first}(G \Rightarrow G, X \Rightarrow \text{stk})$$
- An unwanted occurrence of a concrete port graph G in the system:
  $$(G \Rightarrow \text{stk})$$
- Instead of yielding failure stk, the problem can be “repaired” by inserting the necessary rules or strategies in the system in case of failure.

Application : embedding invariant verification

An invariant property is encoded as a special rule in the biochemical program modelling the system and such rule is dynamically checked at each execution step.

- An invariant of the system : $G \Rightarrow G$.
  The strategy verifying such an invariant:
  $$\texttt{first}(G \Rightarrow G, X \Rightarrow \text{stk})$$
- An unwanted occurrence of a concrete port graph G in the system:
  $$(G \Rightarrow \text{stk})$$
- Instead of yielding failure stk, the problem can be “repaired” by inserting the necessary rules or strategies in the system in case of failure.

Application : embedding invariant verification

An invariant property is encoded as a special rule in the biochemical program modelling the system and such rule is dynamically checked at each execution step.

- An invariant of the system : $G \Rightarrow G$.
  The strategy verifying such an invariant:
  $$\texttt{first}(G \Rightarrow G, X \Rightarrow \text{stk})$$
- An unwanted occurrence of a concrete port graph G in the system:
  $$(G \Rightarrow \text{stk})$$
- Instead of yielding failure stk, the problem can be “repaired” by inserting the necessary rules or strategies in the system in case of failure.

ISR 2014
Strategies

Hélène KIRCHNER
Inria

August 2014

Strategy Semantics: Abstract Reduction Systems

Abstract Reduction Systems

Another approach to talk about derivations

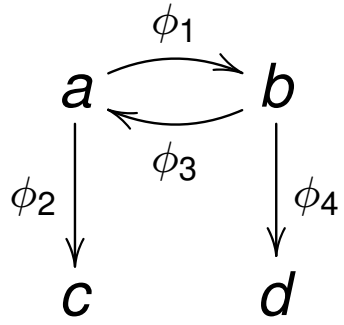
An *Abstract Reduction System (ARS)* is a labelled oriented graph $(\mathcal{O}, \mathcal{S})$ with a set of labels $\mathcal{L}$.
The nodes in $\mathcal{O}$ are called *objects*

The oriented labelled edges in $\mathcal{S}$ are called *steps* :

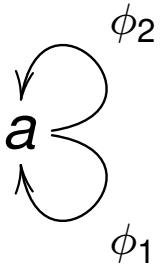
$a \xrightarrow{\phi} b$ or (a, ϕ, b), with *source a*, *target b* and *label* ϕ.

Example:

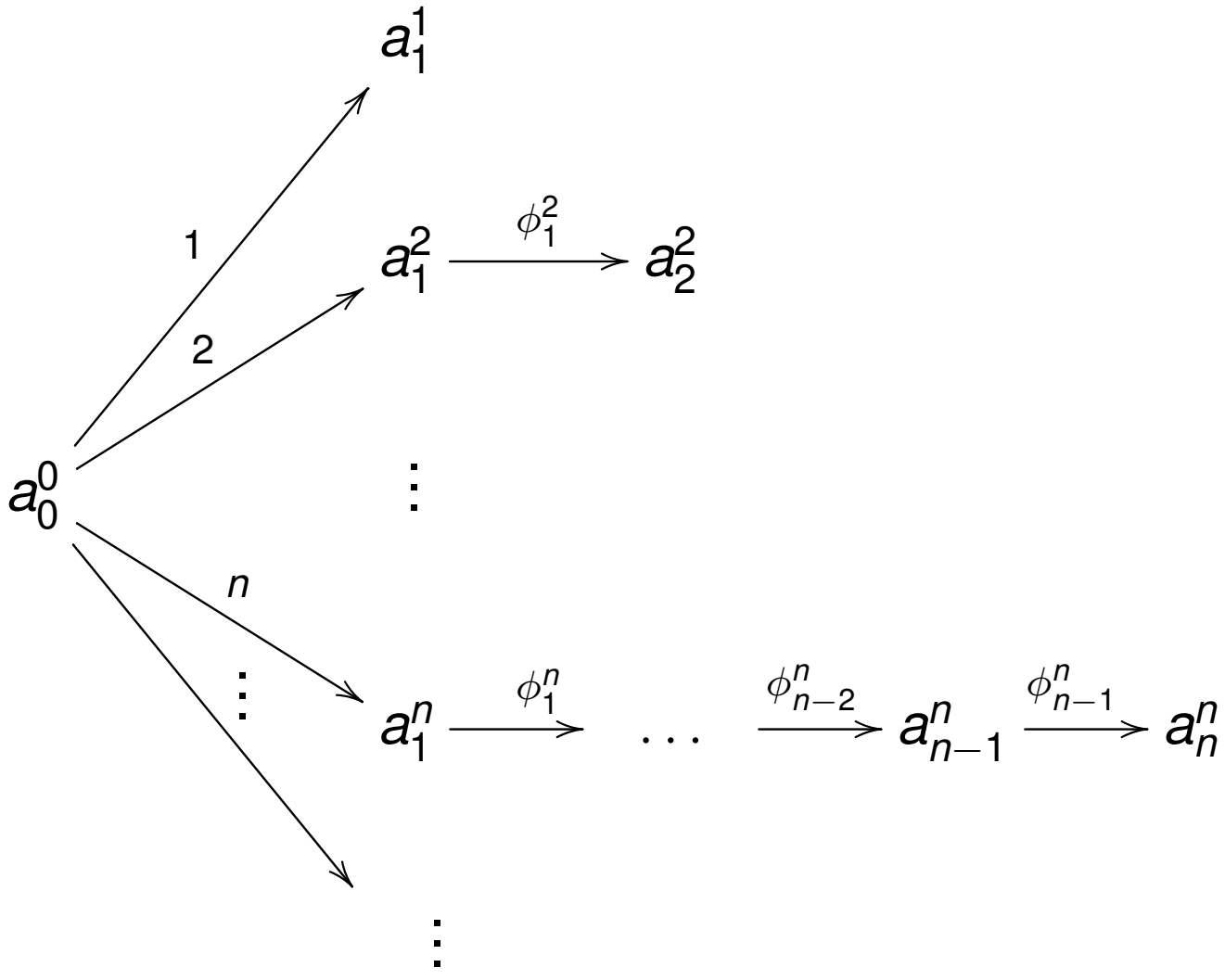
1. $\mathcal{A}_{lc} =$



2. $\mathcal{A}_c =$



Another example of ARS



Derivations

For a given ARS $\mathcal{A}$:

1. *$\mathcal{A}$-derivation*:

$$\pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n$$

or

$$a_0 \xrightarrow{\pi} a_n$$

The *source* of π is a_0 and $Dom(\pi) = \{a_0\}$.
The *target* of π is a_n and $\pi \bullet a_0 = \{a_n\}$.

The *length* of π, denoted $|\pi|$ is n and a step is of length 1. A derivation is empty when its length is 0 and its source and target are the same.

2. The set of all derivations is denoted $\mathcal{D}(\mathcal{A})$.

Derivations

For a given ARS $\mathcal{A}$:

1. *$\mathcal{A}$-derivation*:

$$\pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n$$

or

$$a_0 \xrightarrow{\pi} a_n$$

The *source* of π is a_0 and $Dom(\pi) = \{a_0\}$.
The *target* of π is a_n and $\pi \bullet a_0 = \{a_n\}$.
The *length* of π, denoted $|\pi|$ is n and a step is of length 1. A derivation is empty when its length is 0 and its source and target are the same.

2. The set of all derivations is denoted $\mathcal{D}(\mathcal{A})$.

Abstract strategies

For a given ARS $\mathcal{A}$:

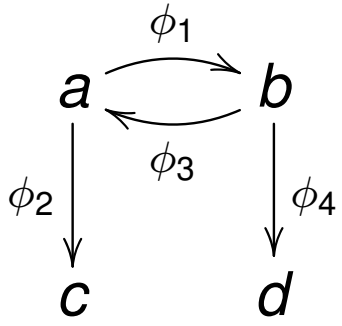
1. An *abstract strategy* ζ is a subset of non-empty derivations of the set of all derivations (finite or not) of $\mathcal{A}$.
2. $Dom(\zeta) = \bigcup_{\pi \in \zeta} Dom(\pi)$
3. $\zeta \bullet a = \{b \mid \exists \pi \in \zeta \text{ such that } a \xrightarrow{\pi} b\} = \{\pi \bullet a \mid \pi \in \zeta\}$.
4. ζ *is undefined (fails)* on a if $a \notin Dom(\zeta)$.
5. ζ is indeterminate on a if $a \in Dom(\zeta)$ and there is only infinite derivations in ζ of source a.
6. ζ normalizes a if $a \in Dom(\zeta)$ and if there is no infinite derivation in ζ of source a.

Abstract strategies

For a given ARS $\mathcal{A}$:

1. An *abstract strategy* ζ is a subset of non-empty derivations of the set of all derivations (finite or not) of $\mathcal{A}$.
2. $Dom(\zeta) = \bigcup_{\pi \in \zeta} Dom(\pi)$
3. $\zeta \bullet a = \{b \mid \exists \pi \in \zeta \text{ such that } a \xrightarrow{\pi} b\} = \{\pi \bullet a \mid \pi \in \zeta\}$.
4. ζ *is undefined (fails)* on a if $a \notin Dom(\zeta)$.
5. ζ is indeterminate on a if $a \in Dom(\zeta)$ and there is only infinite derivations in ζ of source a.
6. ζ normalizes a if $a \in Dom(\zeta)$ and if there is no infinite derivation in ζ of source a.

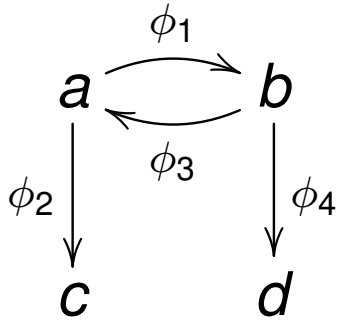
Examples

$\mathcal{A}_{lc} =$ 

A few strategies:

1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$, $\zeta_1 \bullet a = \{a, b, c, d\} = \zeta_1 \bullet b$, $\zeta_1 \bullet c = \zeta_1 \bullet d = \varnothing$.
2. $\zeta_2 = \varnothing$, for all x in $\mathcal{O}_{lc}$, $\zeta_2 \bullet x = \varnothing$.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$,
   a always converges to c: $\zeta_3 \bullet a = \{c\}$;
   b is not transformed (as well as c and d): $\zeta_3 \bullet b = \varnothing$.

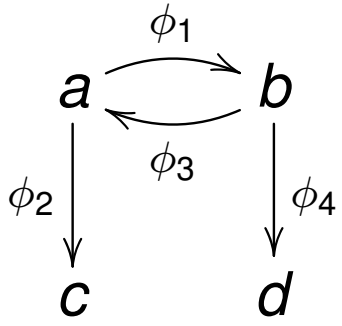
Examples

$\mathcal{A}_{lc} =$ 

A few strategies:

1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$, $\zeta_1 \bullet a = \{a, b, c, d\} = \zeta_1 \bullet b$, $\zeta_1 \bullet c = \zeta_1 \bullet d = \varnothing$.
2. $\zeta_2 = \varnothing$, for all x in $\mathcal{O}_{lc}$, $\zeta_2 \bullet x = \varnothing$.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$,
   a always converges to c: $\zeta_3 \bullet a = \{c\}$;
   b is not transformed (as well as c and d): $\zeta_3 \bullet b = \varnothing$.

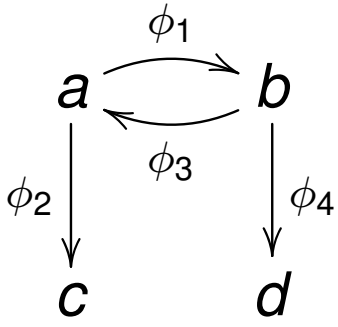
Examples

$\mathcal{A}_{lc} =$ 

A few strategies:

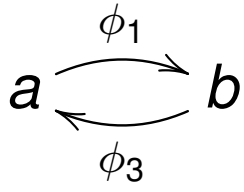
1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$, $\zeta_1 \bullet a = \{a, b, c, d\} = \zeta_1 \bullet b$, $\zeta_1 \bullet c = \zeta_1 \bullet d = \varnothing$.
2. $\zeta_2 = \varnothing$, for all x in $\mathcal{O}_{lc}$, $\zeta_2 \bullet x = \varnothing$.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$,
   a always converges to c: $\zeta_3 \bullet a = \{c\}$;
   b is not transformed (as well as c and d): $\zeta_3 \bullet b = \varnothing$.

Some consequences of definitions

- $\mathcal{A}_{lc} =$ 

Let ζ be defined as $\{a \xrightarrow{\phi_1} b \xrightarrow{\phi_4} d\}$

  ζ does not normalize b.

- $\mathcal{A}_{loop} =$ 

  Let ζ be the subset of derivations of length 2. ζ normalizes a and b.

Suitable properties

- ζ is factor-closed (resp. prefix-closed) iff for any derivation $\pi \in \zeta$, any factor (resp. prefix) of π is also in ζ.
- ζ is closed by composition iff for any two composable derivations $\pi, \pi' \in \zeta$, their composition $\pi; \pi'$ is in ζ too.
- *Prefix saturation of strategies*:

  for any derivation $\pi = a_0 \xrightarrow{\phi_0} a_1 \ldots \xrightarrow{\phi_{n-1}} a_n \in \zeta$,

  let $\bar{\pi}$ the set of all prefix derivations $\pi = a_0 \xrightarrow{\phi_0} a_1 \ldots \xrightarrow{\phi_{k-1}} a_k$ for $1 \leq k \leq n$.

  Let $\bar{\zeta}$ be the prefix saturation of the abstract strategy ζ defined as

$$\bar{\zeta} = \{\bar{\pi} \mid \pi \in \zeta\}$$

Are we happy with the definition of abstract strategies?

Pros a general concept that covers several areas: reduction systems and deduction systems as explored in [KKK08], programming strategies,...

Cons usual notions may be counter-intuitive, extensional, memory-less, without look-ahead,...

Other approaches : “intensional” definitions ([Dan Dougherty]).
The goal is

1. to build strategic derivations “step by step”
2. to take into account the history at each step

Strategies with memory - traced objects

A *traced-object* is pair $[\alpha]\, a$ where α is a sequence of elements of $\mathcal{O} \times \mathcal{L}$ called *trace* or *history*.

$\mathcal{O}^{[\mathcal{A}]}$ is the set of all traced-objects over $\mathcal{A}$:

$$[\alpha]\, a = [(a_0, \phi_0), \ldots, (a_n, \phi_n)]\, a$$

each object a memorizes how it has been reached with the trace α

Intensional strategies

Given an history and a object, determine the possible next steps.

An *intensional strategy* for $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ is a partial function

$$\begin{aligned} \lambda : \quad & \mathcal{O}^{[\mathcal{A}]} \mapsto 2^{\mathcal{S}} \\ & [(a_0, \phi_0), \ldots, (a_n, \phi_n)]a \mapsto \{(a, \phi'_1, a'_1) \ldots (a, \phi'_k, a'_k)\} \\ & s.t.\ \{(a, \phi'_1, a'_1) \ldots (a, \phi'_k, a'_k)\} \subseteq \{\pi \in \mathcal{S} \mid Dom(\pi) = a\} \end{aligned}$$

If $\lambda([\alpha]\, a) = \varnothing$, then a is a λ-normal form.
If $\lambda([\alpha]\, a)$ is a singleton, then the reduction step under ζ is deterministic.

Intensional strategies

Given an history and a object, determine the possible next steps.

An *intensional strategy* for $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ is a partial function

$$\begin{aligned} \lambda : \quad & \mathcal{O}^{[\mathcal{A}]} \mapsto 2^{\mathcal{S}} \\ & [(a_0, \phi_0), \ldots, (a_n, \phi_n)]a \mapsto \{(a, \phi'_1, a'_1) \ldots (a, \phi'_k, a'_k)\} \\ & s.t.\ \{(a, \phi'_1, a'_1) \ldots (a, \phi'_k, a'_k)\} \subseteq \{\pi \in \mathcal{S} \mid Dom(\pi) = a\} \end{aligned}$$

If $\lambda([\alpha]\, a) = \varnothing$, then a is a λ-normal form.
If $\lambda([\alpha]\, a)$ is a singleton, then the reduction step under ζ is deterministic.

Extensional versus intensional strategies

An intensional strategy λ generates an abstract strategy, called its *extension* ζ_λ:

$$\forall n \in \mathbb{N}, \pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n \in \zeta_\lambda$$

$$\text{iff} \quad \forall j \in [0, n], \quad (a_j \xrightarrow{\phi_j} a_{j+1}) \in \lambda([\alpha]\, a_j)$$

ζ_λ may contain infinite derivations and is closed under taking prefixes.

Extensional versus intensional strategies

An intensional strategy λ generates an abstract strategy, called its *extension* ζ_λ:

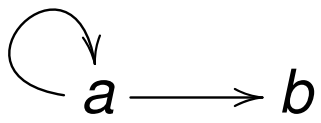
$$\forall n \in \mathbb{N}, \pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n \in \zeta_\lambda$$

$$\text{iff} \quad \forall j \in [0, n], \quad (a_j \xrightarrow{\phi_j} a_{j+1}) \in \lambda([\alpha]\, a_j)$$

ζ_λ may contain infinite derivations and is closed under taking prefixes.

Extensional versus intensional strategies

Expressiveness of intensional strategies, closure properties studied in [BCDK-WRS09].
Extensional more expressive than intensional :



Let ζ be the set of all the reductions which eventually fire the rule $a \to b$:
$\zeta = (a \to a)^*(a \to b)$
there is no intensional strategy λ such that $\zeta_\lambda = \zeta$.

Characteristic property of an intensional strategy

Given an intensional strategy λ for $\mathcal{A}$, to characterize "next" steps, we are left to decide whether

$$\pi_i = a_0 \xrightarrow{\phi_0} a_1 \dots a_{n-1} \xrightarrow{\phi_{n-1}} a_n \xrightarrow{\phi_n} a \xrightarrow{\phi} a' \in \bar{\zeta_\lambda}$$

Associate to λ its *characteristic property* $\mathcal{P}_\lambda$ that may depend on the history, on the current step and on future steps:

$$\mathcal{P}_\lambda([\alpha]\, a, \phi) \text{ is true}$$
$$\text{iff}$$
$$[\alpha]\, a; a \xrightarrow{\phi} a' \in \bar{\zeta_\lambda}$$

Examples

- $\lambda = \mathit{Innermost}$:
  $\mathcal{P}_{\mathit{Innermost}}([\alpha]\, t, (p, \gamma, \sigma)) : t \xrightarrow{(p', \gamma', \sigma')} t'' \in \mathcal{S} \Rightarrow p \not<_{\mathit{pref}} p'$
- $\lambda = \mathit{ltk}$ is the set of derivations of length k:
  $\mathcal{P}_{\mathit{ltk}} : (|\alpha| < k)$
- $\lambda = \mathcal{R}_1; \mathcal{R}_2$ alternates reduction in $\mathcal{R}_1$ and in $\mathcal{R}_2$:

$$[\alpha]\, t \xrightarrow{\gamma} [\alpha']\, t' \quad \text{iff} \quad \begin{array}{l} t \xrightarrow{\gamma} t' \wedge \\ \alpha = \alpha'.(u, \gamma') \wedge \\ \gamma' \in \mathcal{R}_1 \Rightarrow \gamma \in \mathcal{R}_2 \\ \vee \\ \gamma' \in \mathcal{R}_2 \Rightarrow \gamma \in \mathcal{R}_1 \end{array}$$

ISR 2014
Strategies

Hélène KIRCHNER
Inria

August 2014

Termination and Confluence under strategy

Properties

Strategic rewriting needs careful definitions of

- Termination under strategy
- Normal form under strategy
- Confluence under strategy

Studied in [CFHKirchner-08]

Derivations in Abstract Reduction Systems

For a given ARS $\mathcal{A}$:

1. *$\mathcal{A}$-derivation*: $\pi : a_0 \xrightarrow{\phi_0} a_1 \xrightarrow{\phi_1} a_2 \ldots \xrightarrow{\phi_{n-1}} a_n$ or $a_0 \xrightarrow{\pi} a_n$.
   The *source* of π is a_0 and $Dom(\pi) = \{a_0\}$.
   The *target* of π is a_n if $\pi \bullet a_0 = \{a_n\}$.
   The *length* of π, denoted $|\pi|$ is n and a step is of length 1.
2. The *concatenation* of two derivations $\pi_1; \pi_2$ is defined as $a \xrightarrow{\pi_1}_{\mathcal{A}} b \xrightarrow{\pi_2}_{\mathcal{A}} c$ if $\{a\} = Dom(\pi_1)$ and $\pi_1 \bullet a = Dom(\pi_2) = \{b\}$.
   Then $\pi_1; \pi_2 \bullet a = \pi_2 \bullet \pi_1 \bullet a = \{c\}$

Properties : Termination

For a given ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$:

- $\mathcal{A}$ is *terminating* (or *strongly normalizing*) if all its derivations are of finite length;
- An object a in $\mathcal{O}$ is *irreducible* if a is the source of no edge;
- A derivation is *normalizing* when its target is irreducible
- An ARS is *weakly terminating* if every object a is the source of a normalizing derivation.

Properties : Termination

For a given ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$:

- $\mathcal{A}$ is *terminating* (or *strongly normalizing*) if all its derivations are of finite length;
- An object *a* in $\mathcal{O}$ is *irreducible* if *a* is the source of no edge;
- A derivation is *normalizing* when its target is irreducible
- An ARS is *weakly terminating* if every object *a* is the source of a normalizing derivation.

Properties : Termination

For a given ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$:

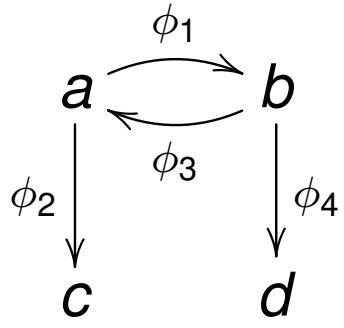
- $\mathcal{A}$ is *terminating* (or *strongly normalizing*) if all its derivations are of finite length;
- An object *a* in $\mathcal{O}$ is *irreducible* if *a* is the source of no edge;
- A derivation is *normalizing* when its target is irreducible
- An ARS is *weakly terminating* if every object *a* is the source of a normalizing derivation.

Termination of abstract strategies

For a given ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ and strategy ζ:

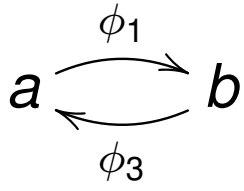
- $\mathcal{A}$ is *ζ-terminating* if all derivations in ζ are of finite length;
- An object *a* is ζ-irreducible if there is no derivation of source *a* in ζ.
- A derivation is *ζ-normalizing* when its target is ζ-irreducible;
- An ARS is *weakly ζ-terminating* if every object *a* is the source of a ζ-normalizing derivation.

Some consequences of definitions

- $\mathcal{A}_{lc} =$ 

Let ζ be defined as $\{a \xrightarrow{\phi_1} b \xrightarrow{\phi_4} d\}$

  b is ζ-irreducible since there is no derivation in ζ with source b.

- $\mathcal{A}_{loop} =$ 

Let ζ be the subset of derivations of length 2. $\mathcal{A}$ is trivially ζ-terminating, while there is no ζ-normalized object.

Properties : Confluence

An ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ is *confluent* if

for all objects a, b, c in $\mathcal{O}$, and all $\mathcal{A}$-derivations π_1 and π_2,
when $a \xrightarrow{\pi_1} b$ and $a \xrightarrow{\pi_2} c$,
there exist d in $\mathcal{O}$ and two $\mathcal{A}$-derivations π_3, π_4 such that
$c \xrightarrow{\pi_3} d$ and $b \xrightarrow{\pi_4} d$.

Confluence of abstract strategies (1)

Weak Confluence under strategy

An ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ is *weakly confluent* under strategy ζ if

for all objects a, b, c in $\mathcal{O}$, and all $\mathcal{A}$-derivations π_1 and π_2 in ζ,
when $a \xrightarrow{\pi_1} b$ and $a \xrightarrow{\pi_2} c$

there exists d in $\mathcal{O}$ and two $\mathcal{A}$-derivations π'_3, π'_4 in ζ such that
$\pi'_3 : a \to b \to d$ and $\pi'_4 : a \to c \to d$.

Confluence of abstract strategies (2)

Strong Confluence under strategy

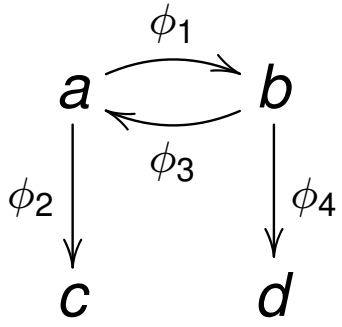
An ARS $\mathcal{A} = (\mathcal{O}, \mathcal{S})$ is *strongly confluent* under strategy ζ if

for all objects a, b, c in $\mathcal{O}$, and all $\mathcal{A}$-derivations π_1 and π_2 in ζ,
when $a \xrightarrow{\pi_1} b$ and $a \xrightarrow{\pi_2} c$

there exists d in $\mathcal{O}$ and two $\mathcal{A}$-derivations π_3, π_4 in ζ such that:

1. $b \xrightarrow{\pi_3} d$ and $c \xrightarrow{\pi_4} d$;
2. $\pi_1; \pi_3$ and $\pi_2; \pi_4$ belong to ζ.

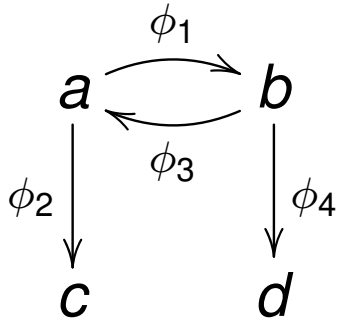
Example

$\mathcal{A}_{lc} =$ 

Consider the following various strategies:

1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$: $\mathcal{A}_{lc}$ is neither weakly nor strongly confluent under ζ_1: $\pi_1 : a \xrightarrow{\phi_1} b \xrightarrow{\phi_4} d$ and $\pi_2 : a \xrightarrow{\phi_2} c$.
2. $\zeta_2 = \varnothing$: $\mathcal{A}_{lc}$ is trivially both weakly and strongly confluent under ζ_2.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$: $\mathcal{A}_{lc}$ is weakly and strongly confluent under ζ_3.

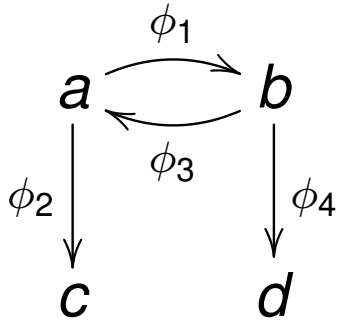
Example

$\mathcal{A}_{lc} =$ 

Consider the following various strategies:

1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$: $\mathcal{A}_{lc}$ is neither weakly nor strongly confluent under ζ_1: $\pi_1 : a \xrightarrow{\phi_1} b \xrightarrow{\phi_4} d$ and $\pi_2 : a \xrightarrow{\phi_2} c$.
2. $\zeta_2 = \varnothing$: $\mathcal{A}_{lc}$ is trivially both weakly and strongly confluent under ζ_2.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$: $\mathcal{A}_{lc}$ is weakly and strongly confluent under ζ_3.

Example

$\mathcal{A}_{lc} =$ 

Consider the following various strategies:

1. $\zeta_1 = \mathcal{D}(\mathcal{A}_{lc})$: $\mathcal{A}_{lc}$ is neither weakly nor strongly confluent under ζ_1: $\pi_1 : a \xrightarrow{\phi_1} b \xrightarrow{\phi_4} d$ and $\pi_2 : a \xrightarrow{\phi_2} c$.
2. $\zeta_2 = \varnothing$: $\mathcal{A}_{lc}$ is trivially both weakly and strongly confluent under ζ_2.
3. $\zeta_3 = \{(\phi_1\phi_3)^*\phi_2\}$: $\mathcal{A}_{lc}$ is weakly and strongly confluent under ζ_3.

Examples

Exercise: For $\mathcal{A} = a \xrightarrow{\phi_1} b \xrightarrow{\phi_2} c \xrightarrow{\phi_3} d$
Study termination and confluence under
$\zeta = \{a \xrightarrow{\phi_1} b \xrightarrow{\phi_2} c \xrightarrow{\phi_3} d\}$
$\zeta' = \{a \xrightarrow{\phi_1} b, a \xrightarrow{\phi_1} b \xrightarrow{\phi_2} c \xrightarrow{\phi_3} d\}$
$\zeta'' = \{a \xrightarrow{\phi_1} b, a \xrightarrow{\phi_1} b \xrightarrow{\phi_2} c, a \xrightarrow{\phi_1} b \xrightarrow{\phi_2} c \xrightarrow{\phi_3} d\}$
$\zeta'' = \bar{\zeta}$ - prefix-closure of ζ
ζ''' - factor-closure of ζ.

Other direct techniques

Specific techniques

Confluence, termination, completeness under traversal strategies:
based on schematization of derivation trees
([GK-TOCL09])

Termination of rewriting under innermost, outermost, lazy strategies
([Giesl,Middeldorp])

Strategies Transformation to equivalent rewrite systems
([FGK-PPDP-2003,Moreau&all])