Specifying Complex and Structured Systems with Evolving Algebras

Wolfgang May

Institut für Informatik

Universität Freiburg

Germany

Evolving Algebras

- Formal specification method for describing operational semantics.
- Since recently aka
  Gurevich Abstract State Machines.
- Transition systems;
- States are *static algebras*, i.e., first-order interpretations over a *functional* signature Σ: $\mathcal{A} = (A, \mathcal{S})$ with $\mathcal{S}$ (*superuniverse*) a set of elements and A an interpretation of the function symbols, $A(f) : \mathcal{S}^{\mathsf{ord}(f)} \to \mathcal{S}$.
  $\mathcal{S}$ contains a distinguished element *undef*.
- An Evolving Algebra $\mathcal{E}$ is given by an initial state $\mathcal{Z}(\mathcal{E})$ and a set $\mathcal{R}(\mathcal{E})$ of transition rules.

Rules

An *elementary update* is an update of the interpretation of a function symbol at one location:

$$u: \ f(t_1, \ldots, t_n) := t_0$$

- If u is an elementary update, then u is a rule.
- If $r_1, \ldots, r_n$ are rules, then $r_1, \ldots, r_n$ is a rule ("block").
- If $g_1, \ldots, g_k$ are boolean terms over Σ ("guards") and $r_1, \ldots, r_{k+1}$ are rules, then
  $r =$
  if g_1 then r_1
  elsif g_2 then r_2 elsif ... elsif g_k then r_k
  else r_{k+1} endif
  is a rule.

A rule schema stands for a block of all its ground instances.

A program is a finite set of rules.

Updates

For an n-ary function symbol $f \in \Sigma$ and $s_1, \ldots, s_n \in \mathcal{S}$, $(f, s_1, \ldots, s_n)$ is a *location* over Σ and $\mathcal{S}$.

An *update* over a signature Σ and a superuniverse $\mathcal{S}$ is a pair (ℓ, s), where ℓ is a location and $s \in \mathcal{S}$.

Let $\mathcal{A}$ be a static algebra.

- For $r \equiv f(t_1, \ldots, t_n) := t_0$,
  $\mathsf{Upd}(r, \mathcal{A}) := \{(f, \mathcal{A}(t_1), \ldots, \mathcal{A}(t_n), \mathcal{A}(t_0))\}$.
- For a block $r \equiv r_1, \ldots, r_n$,
  $\mathsf{Upd}(r, \mathcal{A}) := \mathsf{Upd}(r_1, \mathcal{A}) \cup \ldots \cup \mathsf{Upd}(r_n, \mathcal{A})$.
- For a rule $r \equiv$

  if g_1 then r_1 elsif g_2 then r_2 ... elsif g_k then r_k

  else r_{k+1} endif

  and $\mathcal{A} \models g_i$ and $\mathcal{A} \not\models g_j$ for all $j < i$,
  $\mathsf{Upd}(r, \mathcal{A}) := \mathsf{Upd}(r_i, \mathcal{A})$.
  If $\mathcal{A} \not\models g_j$ for all j, then
  $\mathsf{Upd}(r, \mathcal{A}) := \mathsf{Upd}(r_{k+1}, \mathcal{A})$.
- For a program $P = \{r_1, \ldots, r_n\}$,
  $\mathsf{Upd}(P, \mathcal{A}) := \mathsf{Upd}(r_1, \mathcal{A}) \cup \ldots \cup \mathsf{Upd}(r_n, \mathcal{A})$.

A set U of updates is consistent if for every location ℓ, $|\{s \mid (\ell, s) \in U\}| \leq 1$.

Transitions

Let U be a consistent set of updates and $\mathcal{A}$ a static algebra. Then the state $\mathcal{B} = (B, \mathcal{S})$ obtained by executing U in $\mathcal{A}$ is given as

$$(B(f))(s_1, \ldots, s_n)) = \begin{cases} s & \text{if } (f, s_1, \ldots, s_n, s) \in U \ , \\ (A(f))(s_1, \ldots, s_n) & \text{otherwise} \ . \end{cases}$$

- The static algebra which is obtained by applying a rule $r \in \mathcal{R}(\mathcal{E})$ in a static algebra $\mathcal{A}$ (i.e. executing $\mathsf{Upd}(r, \mathcal{A})$ is denoted by $r(\mathcal{A})$).
- For a set $\mathcal{R}$ of rules, $\mathcal{R}(\mathcal{A})$ denotes the static algebra which is obtained by executing $\mathsf{Upd}(\mathcal{R}, \mathcal{A})$ in $\mathcal{A}$.

$$((r^{part}(\mathcal{A}))(f))(s_1, \ldots, s_n) := \begin{cases} s & \text{if } (f, s_1, \ldots, s_n, s) \in \mathsf{Upd}(r, \mathcal{A}), \\ \mathit{undef} & \text{otherwise} \ . \end{cases}$$

Operations on Static Algebras

$$\begin{aligned}\mathsf{dom}(\mathcal{A}) := \;& \{(f, s_1, \ldots, s_n) \mid f \in \Sigma, s_i \in \mathcal{S}, \\ & (A(f))(s_1, \ldots, s_n) \neq \mathit{undef}\,\}\end{aligned}$$

Let $\mathcal{A} = (A, \mathcal{S})$ and $\mathcal{A}' = (A', \mathcal{S})$ be static algebras over a signature Σ and L a set of locations over Σ and $\mathcal{S}$.

Union $\mathcal{B} := \mathcal{A} \cup \mathcal{A}'$:

$$\begin{aligned}&(B(f))(s_1, \ldots, s_{\mathsf{ord}(f)}) := \\ &\quad \sqcup((A(f))(s_1, \ldots, s_{\mathsf{ord}(f)}), (A'(f))(s_1, \ldots, s_{\mathsf{ord}(f)}))\ .\end{aligned}$$

Restriction of $\mathcal{A}$ *to* L, $\mathcal{A}|_L$:

$$(A|_L\ (f))(s_1, \ldots, s_n) := \begin{cases} (A(f))(s_1, \ldots, s_n) & \text{if } (f; s_1, \ldots, s_n) \in L\ , \\ \mathit{undef} & \text{otherwise}\ . \end{cases}$$

Superposition $\mathcal{B} = \mathcal{A} \uplus_L \mathcal{A}'$ of $\mathcal{A}$ with $\mathcal{A}'$ on L:

$$(B(f))(s_1, \ldots, s_n) := \begin{cases} (A'(f))(s_1, \ldots, s_n) & \text{if } (f; s_1, \ldots, s_n) \in L\ , \\ (A(f))(s_1, \ldots, s_n) & \text{otherwise}\ . \end{cases}$$

As a shorthand, $\mathcal{A} \uplus \mathcal{A}'$ stands for $\mathcal{A} \uplus_{\mathsf{dom}(\mathcal{A}')} \mathcal{A}'$.

Theorem 1

Let $\mathcal{A}$ a static algebra, and $\mathcal{R}$ a set of rules. Then

a) $\mathcal{R}$ is consistently applicable in $\mathcal{A}$ iff

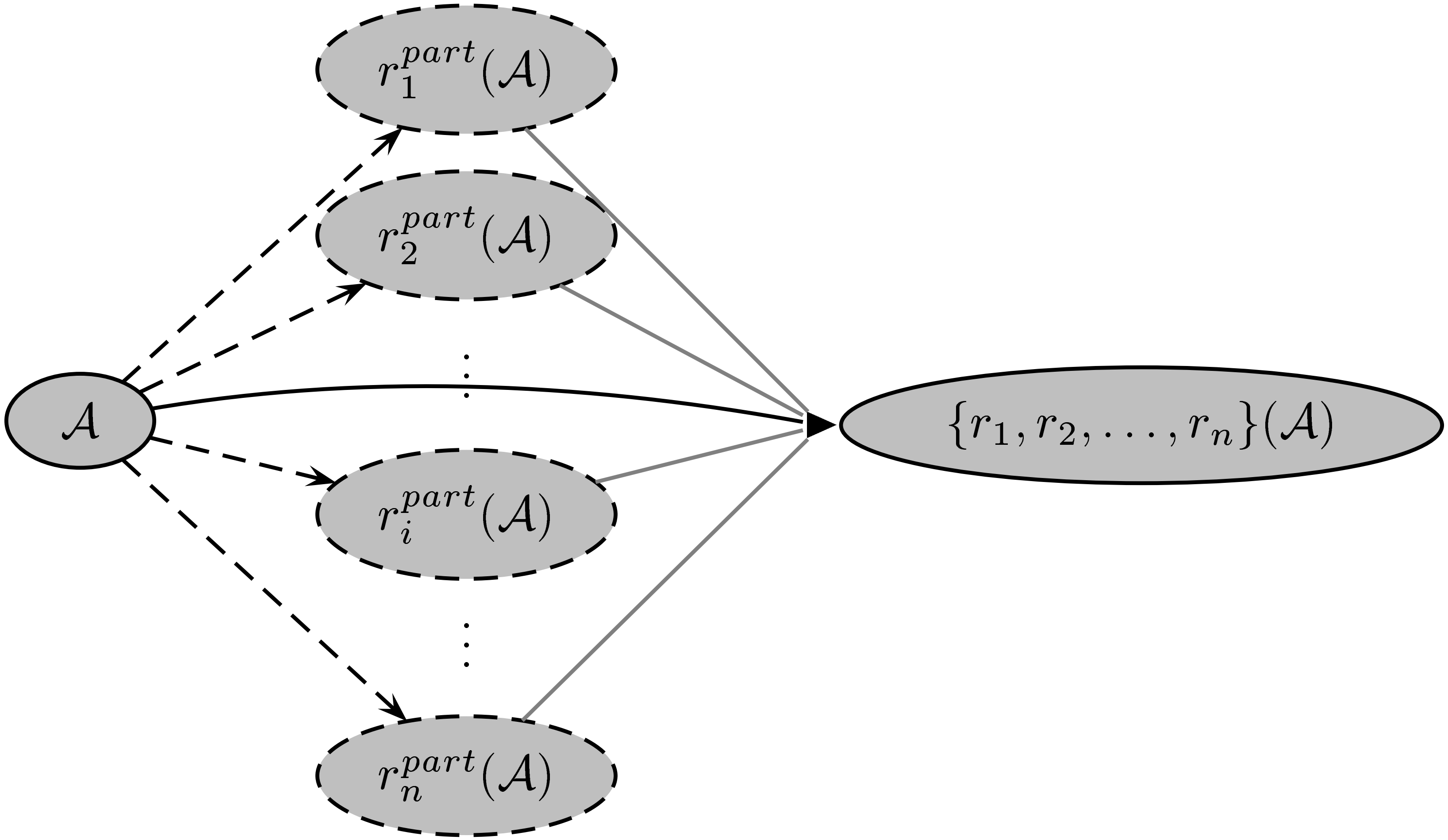
$$\bigcup_{r \in \mathcal{R}} r^{part}(\mathcal{A})$$

is consistent.

b) If $\mathcal{R}$ is consistently applicable in $\mathcal{A}$, then

$$\mathcal{R}(\mathcal{A}) = \mathcal{A} \uplus \bigcup_{r \in \mathcal{R}} r^{part}(\mathcal{A}) \ .$$

State Space with Partial Interpretations

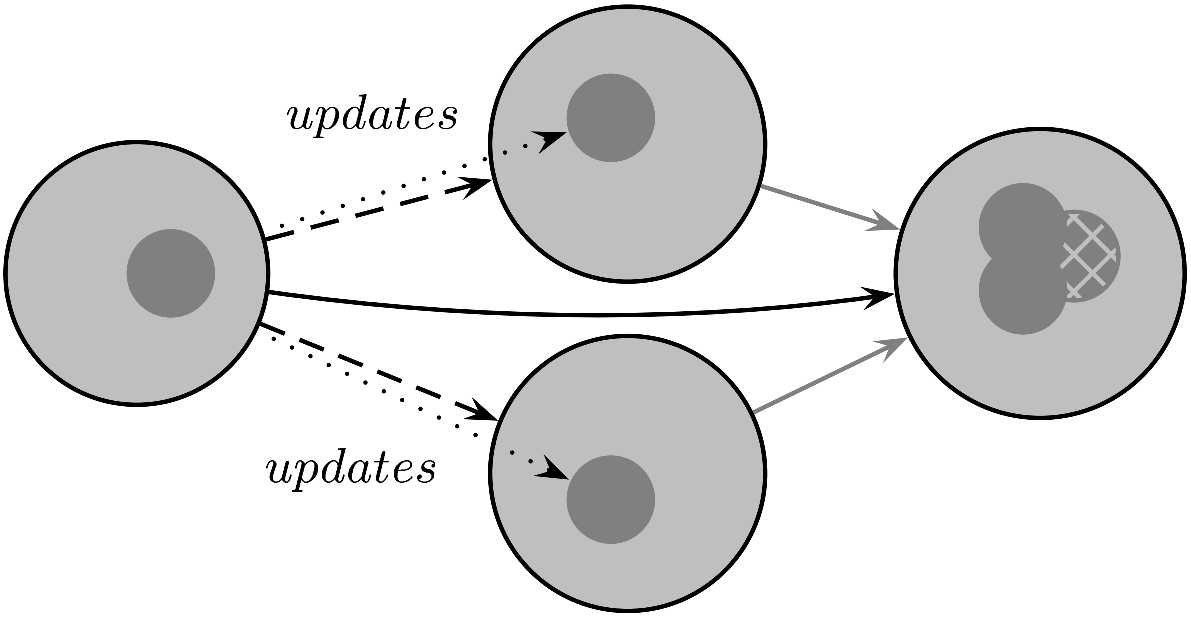


Homogenous State Space

Frame-Problem

Total Interpretations: as-is knowledge

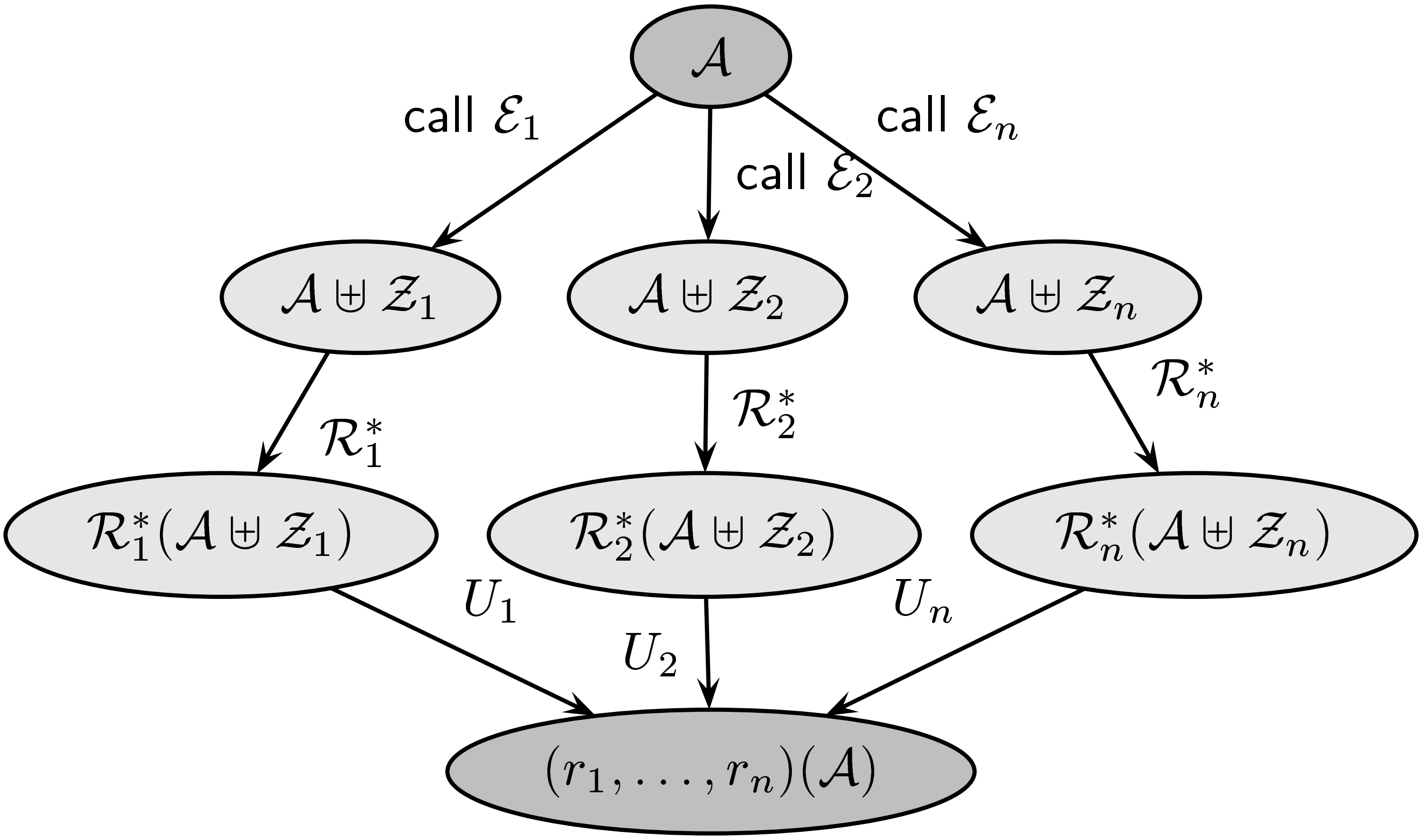
Partial Interpretations: safe, procedure knowledge



Structured Evolving Algebras: Complex Computations Instead of Elementary Rules

$$r_i: \quad \textsf{if } g_i \textsf{ then call } \mathcal{E}_i$$

Initialize $\mathcal{E}_i$ and run $\mathcal{R}(\mathcal{E}_i)$ until a fixpoint is reached; return accumulated net-updates:

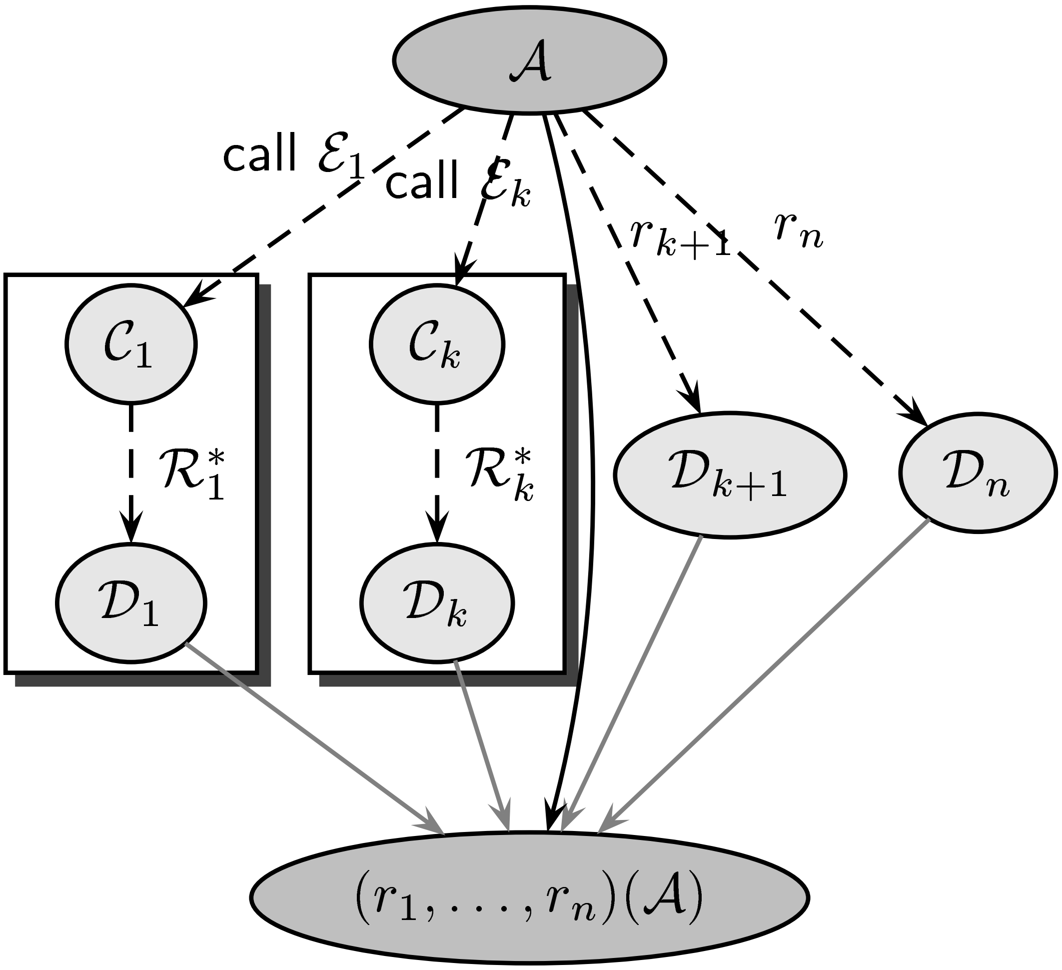


$\mathcal{E}_i = (\mathcal{Z}_i, \mathcal{R}_i)$;
$U_i =$ set of net updates which are executed by running $\mathcal{E}_i$ on $\mathcal{A}$.

Hierarchical State Space

$r_1, \ldots, r_k$: if g_i then call $\mathcal{E}_i$,

$r_{k+1}, \ldots, r_n$ classical EA-Rules.



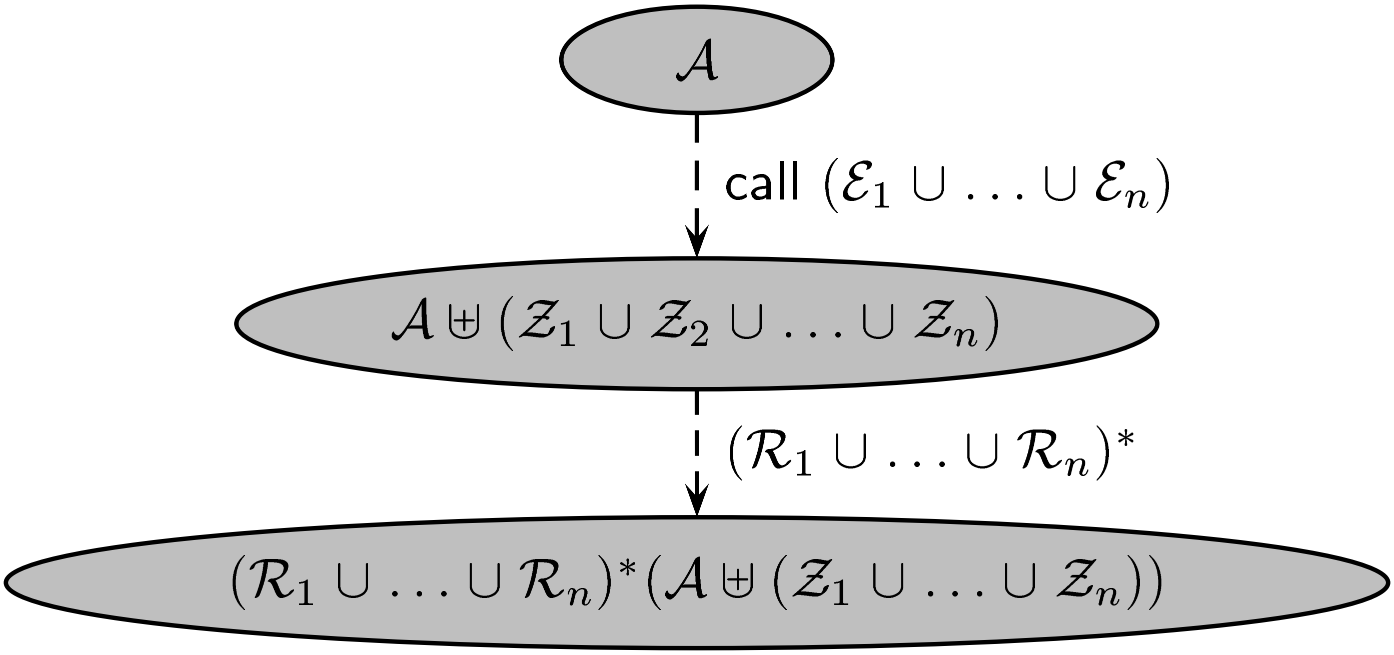
Structured EA's

Based on this concept, arbitrary possibilities for structuring programs and computations can be provided: A static algebra can be given at certain situations to another set of rules.

Encapsulated parallel: Every visible atomic transition is a complete execution of an Evolving Algebra (cf. Slide 10).

Joined parallel:

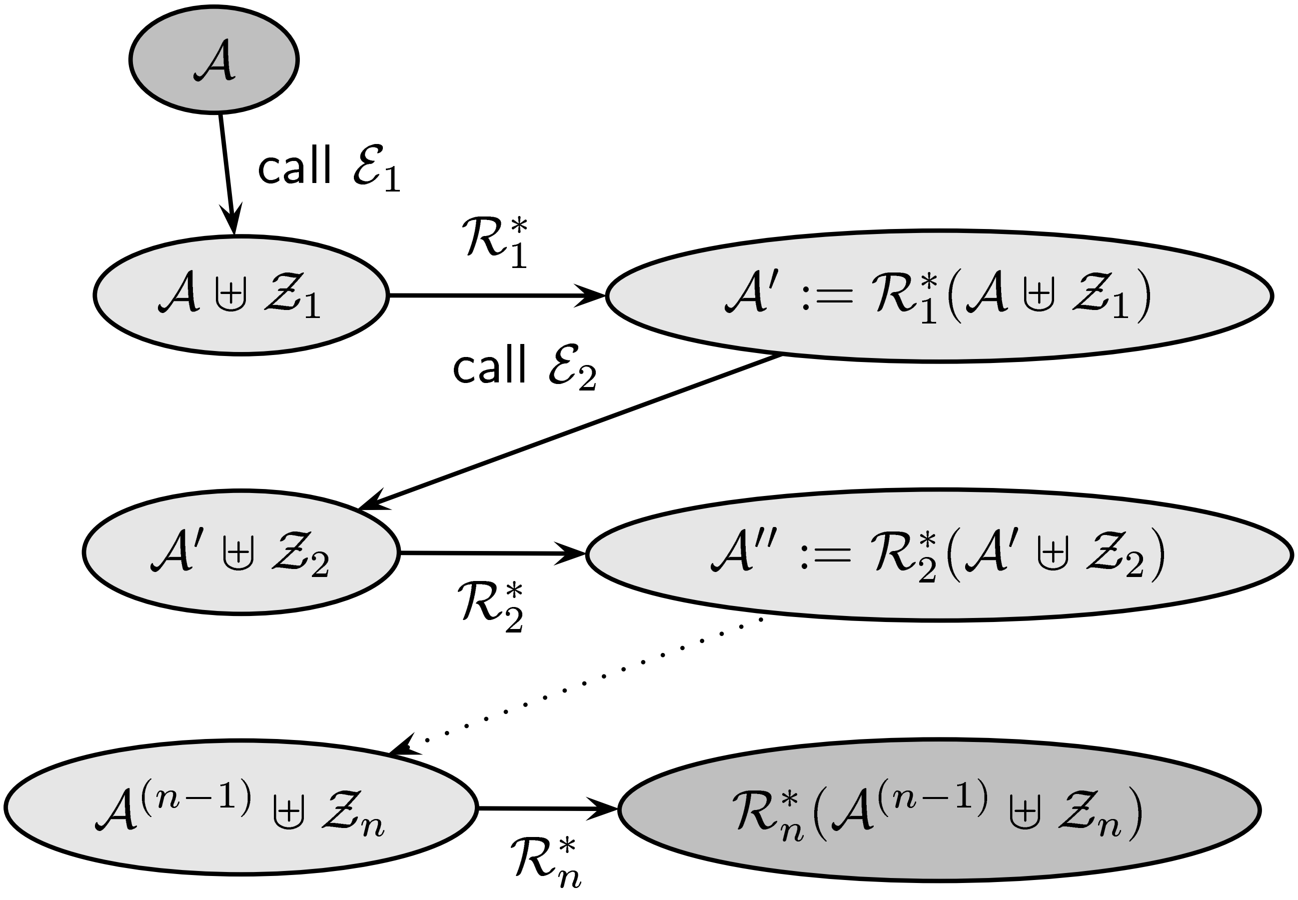
$$(\mathcal{Z}_1, \mathcal{R}_1) \cup \ldots \cup (\mathcal{Z}_1, \mathcal{R}_1) = \\ = (\mathcal{Z}_1 \cup \ldots \cup \mathcal{Z}_n, \mathcal{R}_1 \cup \ldots \cup \mathcal{R}_n) \ .$$



(Encapsulated) Sequential Composition

if g then call $name_1$; call $name_2$; $\ldots$; call $name_n$

Every Evolving Algebra is initialized and executed until it reaches a fixpoint and then the resulting state is given to the next Evolving Algebra:



$$= \mathcal{E}_n(\mathcal{E}_{n-1}(\ldots(\mathcal{E}_1(\mathcal{A}))))$$

Transition Expressions

$\mathbb{R}$: Set of transition expressions:

- If r is a rule, then $\{r\} \in \mathbb{R}$.
- If $\mathcal{A}$ is a static algebra, then $\{\mathcal{A}\} \in \mathbb{R}$.
- If g is a boolean term and $\mathcal{R} \in \mathbb{R}$, then $\{\mathsf{if}\ g\ \ \mathsf{then}\ \ \mathcal{R}\} \in \mathbb{R}$.
- If $\mathcal{Q}, \mathcal{R} \in \mathbb{R}$, then $(\mathcal{Q} \cup \mathcal{R})$, $(\mathcal{Q} \circ \mathcal{R})$ and $(\mathcal{R}^*)$ are also elements of $\mathbb{R}$.

Every rule set $\mathcal{R}$ is an element of $\mathbb{R}$.

Semantics: Elements of $\mathbb{R}$ act as operators on static algebras:

$$\{r\}(\mathcal{A}) := r(\mathcal{A}) \qquad , \qquad \{\mathcal{A}'\}(\mathcal{A}) := \mathcal{A} \uplus_{\mathsf{dom}\mathcal{A}'} \mathcal{A}'$$

$$\{\mathsf{if}\ g\ \ \mathsf{then}\ \ \mathcal{R}\}(\mathcal{A}) := \begin{cases} \mathcal{R}(\mathcal{A}) & \text{if } \mathcal{A} \models g \\ \mathcal{A} & \text{otherwise .} \end{cases}$$

$$(\mathcal{Q} \cup \mathcal{R})(\mathcal{A}) :=$$

$$\mathcal{A} \uplus \left(\mathcal{Q}(\mathcal{A}) \mid_{\mathsf{diffs}(\mathcal{A},\mathcal{Q}(\mathcal{A}))} \cup \mathcal{R}(\mathcal{A}) \mid_{\mathsf{diffs}(\mathcal{A},\mathcal{R}(\mathcal{A}))}\right)$$

$$(\mathcal{Q} \circ \mathcal{R})(\mathcal{A}) := \mathcal{Q}(\mathcal{R}(\mathcal{A}))$$

$$(\mathcal{R}^n)(\mathcal{A}) := (\mathcal{R}^{n-1} \circ \mathcal{R})(\mathcal{A})$$

$$(\mathcal{R}^*)(\mathcal{A}) := \lim_{n\to\infty}(\mathcal{R}^n)(\mathcal{A})$$

$\mathbb{E}$: Systems of Evolving Algebras

- If $\mathcal{A}$ is a static algebra, then $\mathcal{A}$ is an expression in $\mathbb{E}$.
- If r is an Evolving Algebra rule, then $\{r\}$ is an expression in $\mathbb{E}$.
- If g is a boolean term and $\mathcal{E} \in \mathbb{E}$, then $\{\textsf{if } g \textsf{ then } \mathcal{E}\}$ is an expression in $\mathbb{E}$.
- If $\mathcal{E}$ and $\mathcal{F}$ are expressions in $\mathbb{E}$, then $(\mathcal{E} \cup \mathcal{F})$, $(\mathcal{F} \circ \mathcal{E})$, $(\mathcal{F} \bullet \mathcal{E})$, $(\mathcal{F} \prec \mathcal{E})$, $(\mathcal{F} \lessdot \mathcal{E})$, $(\mathcal{E}^{+})$, and $(\mathcal{E}^{*})$ are expressions in $\mathbb{E}$.

$\mathcal{A}$, $\mathcal{R}$**:** Base cases.

$\{\textsf{if } g \textsf{ then } \mathcal{E}\}$**:** execute $\mathcal{E}$ if g is satisfied in the current state.

$\mathcal{E} \cup \mathcal{F}$: (union): The initialization results from the initializations of both subsystems. The system uses the rules from both systems.

$\mathcal{F} \circ \mathcal{E}$: (sequencing): There is no initialization. Each visible action consists of initializing and executing $\mathcal{E}$ followed by initialization and executing $\mathcal{F}$.

$\mathcal{F} \bullet \mathcal{E}$: (prefixing $\mathcal{F}$ with $\mathcal{E}$): $\mathcal{E}$ is completely added to the initialization which then consists of executing $\mathcal{E}$ and initializing $\mathcal{F}$: $\mathcal{F} \bullet \mathcal{E} = \mathcal{F} \bullet \mathcal{E}^*$. The visible actions are the actions of $\mathcal{F}$.

! The construction $\mathcal{O} \bullet \mathcal{E}$ can be used to hide all activities of $\mathcal{E}$ and make only its final state $\mathcal{E}^\infty$ visible.

$\mathcal{E}^+$: (external fixpoint): There is no initialization. Each visible action consists of initializing $\mathcal{E}$ and applying the rules of $\mathcal{E}$ until a fixpoint is reached.

$\mathcal{E}^*$: (internal fixpoint): The initialization consists of the initialization of $\mathcal{E}$. Each visible action consists of applying the rules of $\mathcal{E}$ until a fixpoint is reached.

Semantics

$\mathcal{I} : \mathbb{E} \to \mathbb{E}$, $\mathcal{P} : \mathbb{E} \to \mathbb{R}$ and $\mathcal{Q} : \mathbb{E} \to \mathbb{R}$:

$\mathcal{I}$: Reaching the initial configuration: $(\mathcal{I}(\mathcal{E}))^\infty$ is the initial state of $\mathcal{E}$.

$\mathcal{P}$: Description of actions (elementary rules or complex transitions; represented by an expression in $\mathbb{R}$) which can be executed atomically in this system.

$\mathcal{Q}$: The effect of running the system $\mathcal{E}$ in isolation on a given state: starting $\mathcal{E}$ in a state $\mathcal{A}$, it stops in $(\mathcal{Q}(\mathcal{E}))(\mathcal{A})$ or defines an infinite process.

For a system $\mathcal{E}$, the final state is given by the static Algebra $\mathcal{E}^\infty := (\mathcal{Q}(\mathcal{E}))(\mathcal{O})$.

$\mathcal{S}$	$\mathcal{I}(\mathcal{S})$	$\mathcal{P}(\mathcal{S})$	$\mathcal{Q}(\mathcal{S})$
$\mathcal{A}$	$\mathcal{O}$	$\mathsf{init}(\mathcal{A})$	$\mathsf{init}(\mathcal{A})$
$\{r\}$	$\mathcal{O}$	$\{r\}$	$\{r\}^*$
$\{\mathsf{if}\ g\ \mathsf{then}\ \mathcal{E}\}$	$\mathcal{O}$	$\{\mathsf{if}\ g\ \mathsf{then}\ \mathcal{Q}(\mathcal{E})\}$	$\{\mathsf{if}\ g\ \mathsf{then}\ \mathcal{Q}(\mathcal{E})\}^*$
$\mathcal{E} \cup \mathcal{F}$	$(\mathcal{O} \bullet \mathcal{I}(\mathcal{E})) \cup$	$\mathcal{P}(\mathcal{E}) \cup \mathcal{P}(\mathcal{F})$	$(\mathcal{P}(\mathcal{E}) \cup \mathcal{P}(\mathcal{F}))^* \circ (\mathcal{Q}(\mathcal{I}(\mathcal{E})) \cup \mathcal{Q}(\mathcal{I}(\mathcal{F})))$
	$(\mathcal{O} \bullet \mathcal{I}(\mathcal{F}))$		
$\mathcal{F} \circ \mathcal{E}$	$\mathcal{O}$	$(\mathcal{P}(\mathcal{F}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{F}))\circ$	$(\mathcal{P}(\mathcal{F}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{F})) \circ (\mathcal{P}(\mathcal{E}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{E}))$
		$(\mathcal{P}(\mathcal{E}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{E}))$	$= \mathcal{Q}(\mathcal{F}) \circ \mathcal{Q}(\mathcal{E})$
$\mathcal{F} \bullet \mathcal{E}$	$\mathcal{I}(\mathcal{F}) \circ \mathcal{E}$	$\mathcal{P}(\mathcal{F})$	''
$\mathcal{E}^+$	$\mathcal{O}$	$(\mathcal{P}(\mathcal{E}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{E}))$	$(\mathcal{P}(\mathcal{E}^+))^*$
$\mathcal{E}^*$	$\mathcal{I}(\mathcal{E})$	$(\mathcal{P}(\mathcal{E}))^*$	$(\mathcal{P}(\mathcal{E}))^* \circ \mathcal{Q}(\mathcal{I}(\mathcal{E}))$

Properties

The operators possess the following algebraic properties:

Commutativity: The operation $\_ \cup \_$ is commutative.

Idempotency: The operations $\mathcal{S} \cup \_$, $\mathcal{O} \circ \_$, $\_ \circ \mathcal{O}$, $\mathcal{O} \bullet \_$, $\_ \bullet \mathcal{O}$, and $\_^*$ are idempotent.

Implicite Closures:
$\mathcal{F} \circ \mathcal{E} \equiv \mathcal{F}^* \circ \mathcal{E}^*$, $\quad \mathcal{F} \bullet \mathcal{E} \equiv \mathcal{F} \bullet \mathcal{E}^*$.

Associativity: The binary operators $\cup$, $\circ$ and $\bullet$ are associative.

Special properties of $\mathcal{O} \bullet \_$: $\mathcal{I}(\mathcal{O} \bullet \mathcal{S}) \equiv \mathcal{S}$, $\mathcal{P}(\mathcal{O} \bullet \mathcal{S}) \equiv \emptyset$, $\quad \mathcal{Q}(\mathcal{O} \bullet \mathcal{S}) \equiv \mathcal{Q}(\mathcal{S})$.

Apart from the above-mentioned kind of union, where the rules of both systems can be applied, an *encapsulated union* can be defined as $\mathcal{E} \uplus\!\!\!\!\ast\, \mathcal{F} := \mathcal{E}^* \cup \mathcal{F}^*$.

Flat Operational Model: Changing Active Rule Sets

- Simulate calls of subsystems by changing active rule sets.
- Subterms of $\mathcal{Q}(\mathcal{E})$ associated with systems which are called in some state.
- "atomic" parts: classical Evolving Algebras.

$\rightarrow$ Maintain Control information.

The signature is enriched by the unary function $\rho : \mathsf{sid}'s \rightarrow \{\mathsf{started}, \mathsf{waiting}, \mathsf{getreports}, \mathsf{running}, \mathsf{finished}, \mathsf{ended}\}$ with the meaning

- started: the system just has been started. Subsystems have to be started.
- waiting: the system is waiting for results of its subsystems.
- getreports: the results of subsystems are collected, at the same time, the rules of the system are applied.
- running: only for systems whose transitions are classical Evolving Algebra-rules.
- finished: the computation of the system is finished, but the results are not yet given back.
- ended: the system is finished and its results are given back.

The structure of $\mathcal{Q}(\mathcal{E})$ also has to be modeled:

- $\tau : \mathsf{sid}'s \to \{\mathsf{seq}, \mathsf{par}, \mathsf{iterated}, \mathsf{atomic}\}$,
- $\mathsf{is\_subsystem} : \mathsf{sid}'s \to \mathsf{sid}'s$, subsystems of parallel systems,
- $\mathsf{successor} : \mathsf{sid}'s \to \mathsf{sid}'s$, subsystems of sequential systems,
- $\mathsf{startedby} : \mathsf{sid}'s \to \mathsf{sid}'s$, gives the current calling hierarchy.

For different types of subsystems, there are different additional rules.

Transition diagram