Complex Transitions

in

Evolving Algebras

Wolfgang May

Institut für Informatik, Universität Freiburg
Am Flughafen 17, 79110 Freiburg, Germany
may@informatik.uni-freiburg.de

Demons,

Partial Interpretations,

and

Kripke Structures

Workshop "Evolving Algebras"

Eringerfeld Castle/Paderborn

18.5.1996

Parallelism

Context: Real-World Process Modeling, Rule-based Systems

• Every action is modeled by a single EA-rule

$$\textsf{if } G_0 \textsf{ then } U_0 \textsf{ elsif } G_1 \textsf{ then } U_1 \ldots \textsf{ elsif } G_k \textsf{ then } U_k \textsf{ endif}$$

⇒ In general: Several rules whose preconditions are satisfied, but whose updates are contradictory (cf. Protocols; Kermit).

Tutorial and Lipari: All updates whose guards are satisfied have to be executed.

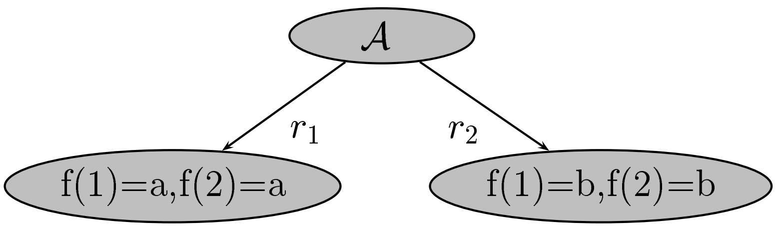
Tutorial: "in case of conflict at any location l, choose the new value for l nondeterministically among all candidate values. It could be called *local* nondeterminism."

Lipari: "To fire an inconsistent update set means to do nothing."

Example:

r_1: f(1):=a, f(2):=a

r_2: f(1):=b, f(2):=b



Possible Outcomes:

Lipari: $\mathcal{A}$ too restrictive

Tutorial: f(1)=a,f(2)=a f(1)=a,f(2)=b

f(1)=b,f(2)=a f(1)=b,f(2)=b

Nondeterminism too much localized

Grained Nondeterminism

Rules have to be executed *atomically* all-or-nothing and a *set* of non-conflicting rules has to be selected for (complete) execution, perhaps rejecting some executable rules.

Tutorial: "Imagine a demon who evaluates all relevant terms in a given state and then makes the necessary changes."

(asynchronous distributed computing): "Instead of one demon, you may have a multitude of demons responsible for different rules".

$\Rightarrow$ Associate one demon with every rule, computing the effects of *his* rule.

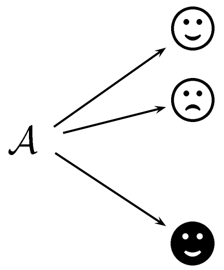
Thus a set of rules corresponds to a *team* of demons:

$\mathcal{A}: \; f(1) = a, \; f(2) = b, \; f(3) = c$

$r_1: \; f(1) := d$

$r_2: \; f(2) := e, f(3) := f$

$r_3: \; f(2) := g$

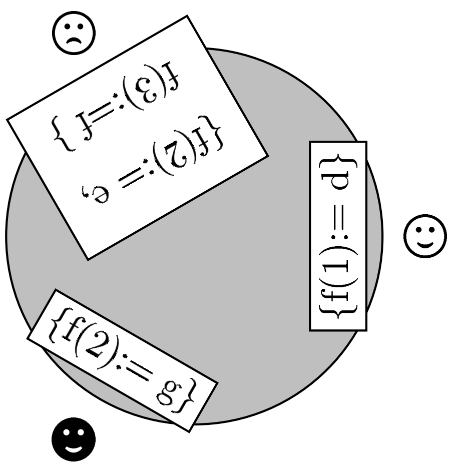


$: \quad r_1 \rightsquigarrow U_1 = \{f(1) := d\}$

$: \quad r_2 \rightsquigarrow U_2 = \{f(2) := e, \; f(3) := f\}$

$: \quad r_3 \rightsquigarrow U_3 = \{f(2) := g\}$

For negotiating a set of non-conflicting rules, the demons sit at a round table, each of them either realizing all his updates or none of them:



Possible Outcomes:

f(1):=d	(r_1)
f(1):=d, f(2):=e, f(3):=f	(r_1, r_2)
f(1):=d, f(2):=g	(r_1, r_3)
f(2):=e, f(3):=f	(r_2)
f(2):=g	(r_3)

$\Rightarrow$ Conflict resolving strategies

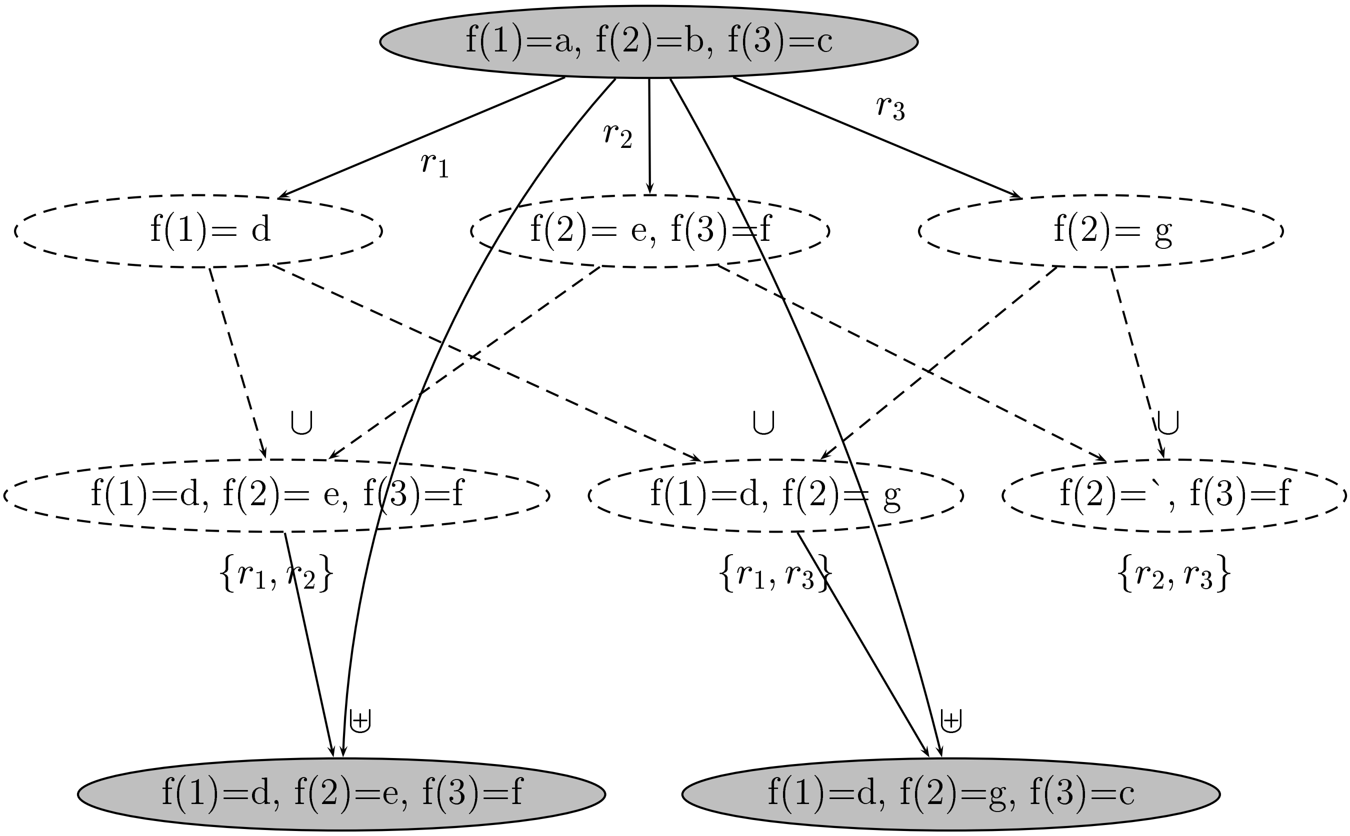
Formalization

$\mathcal{R} := \{r_1, \ldots, r_n\}$ a set of rules, $\mathcal{A}$ a state:

$$\mathsf{Updates}(\mathcal{R}, \mathcal{A}) := \mathsf{Updates}(r_1, \mathcal{A}) \cup \ldots \cup \mathsf{Updates}(r_n, \mathcal{A}) \quad .$$

$$\mathsf{Possible}(\mathcal{R}, \mathcal{A}) := \{\mathsf{Updates}(\mathcal{R}', \mathcal{A}) \mid \mathcal{R}' \subseteq \mathcal{R} \text{ and } \mathsf{Updates}(\mathcal{R}', \mathcal{A}) \text{ is consistent}\} \quad .$$

Partial Interpretations



Formalization

For a set $\mathcal{R}$ of rules:

$$(\mathcal{R}(\mathcal{A}))(f(u_1,\ldots,u_{\mathsf{ord}(f)})) := \begin{cases} u & \text{if } (f,u_1,\ldots,u_{\mathsf{ord}(f)},u) \in \text{Updates}(\mathcal{R}(\mathcal{A})) \\ \mathcal{A}(f(u_1,\ldots,u_{\mathsf{ord}(f)})) & \text{otherwise .} \end{cases}$$

$$\mathsf{writes}(\mathcal{A},r) := \{\ (f;u_1,\ldots,u_{\mathsf{ord}(f)}) : r \text{ updates } f(u_1,\ldots,u_{\mathsf{ord}(f)}) \text{ when applied in } \mathcal{A}\}$$

$$\mathsf{writes}(\mathcal{A},\{r_1,r_2\}) := \mathsf{writes}(\mathcal{A},r_1) \cup \mathsf{writes}(\mathcal{A},r_2)$$

$$(\mathcal{A}|_L)(f(u_1,\ldots,u_{\mathsf{ord}(f)})) := \begin{cases} \mathcal{A}(f(u_1,\ldots,u_{\mathsf{ord}(f)})) & \text{if } (f;u_1,\ldots,u_{\mathsf{ord}(f)}) \in L, \\ \perp & \text{otherwise .} \end{cases}$$

$$(\mathcal{A} \cup \mathcal{A}')(f(u_1,\ldots,u_{\mathsf{ord}(f)})) := \sqcup(\mathcal{A}(f(u_1,\ldots,u_{\mathsf{ord}(f)})), \mathcal{A}'(f(u_1,\ldots,u_{\mathsf{ord}(f)})))$$

where $\sqcup(u,\perp) = \sqcup(\perp,u) = u$ and $\sqcup(u_1, u_2 \neq u_1) = `$,

$$(\mathcal{A} \uplus_L \mathcal{A}')(f(u_1,\ldots,u_{\mathsf{ord}(f)})) := \begin{cases} (A'(f(u_1,\ldots,u_{\mathsf{ord}(f)})) & \text{if } (f;u_1,\ldots,u_{\mathsf{ord}(f)}) \in L, \\ (A(f(u_1,\ldots,u_{\mathsf{ord}(f)})) & \text{otherwise .} \end{cases}$$

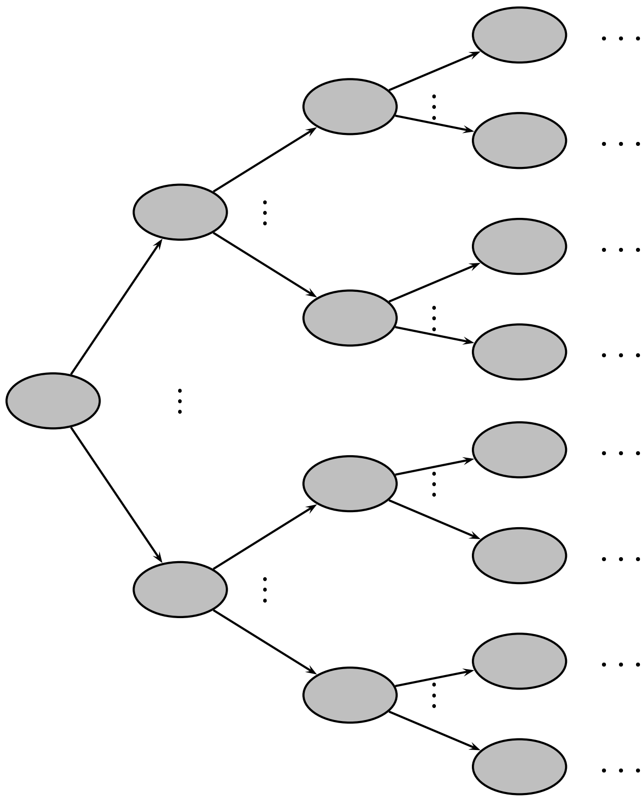
$$\mathcal{A} \uplus \mathcal{A}' := \mathcal{A} \uplus_{\mathsf{dom}(\mathcal{A}')} \mathcal{A}'$$

Result:

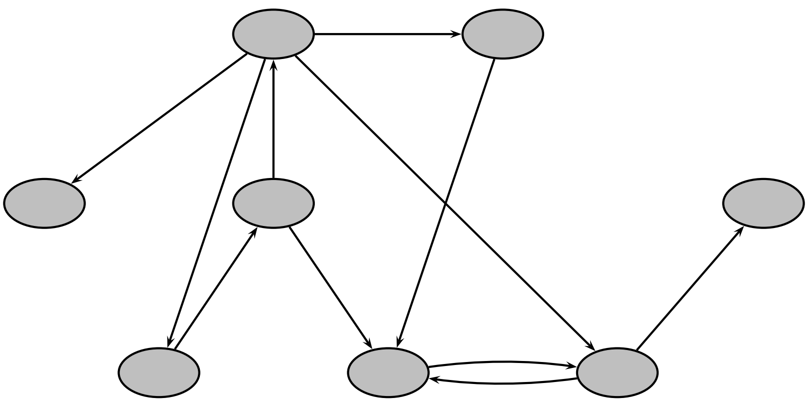
$$\{r_1,r_2\}(\mathcal{A}) = \mathcal{A} \ \uplus_{\mathsf{writes}(\mathcal{A},r_1) \cup \mathsf{writes}(\mathcal{A},r_2)} \ (r_1(\mathcal{A})|_{\mathsf{writes}(\mathcal{A},r_1)} \cup r_2(\mathcal{A})|_{\mathsf{writes}(\mathcal{A},r_1)})$$

Structure

⇒ EA induces a *branching* tree structure:

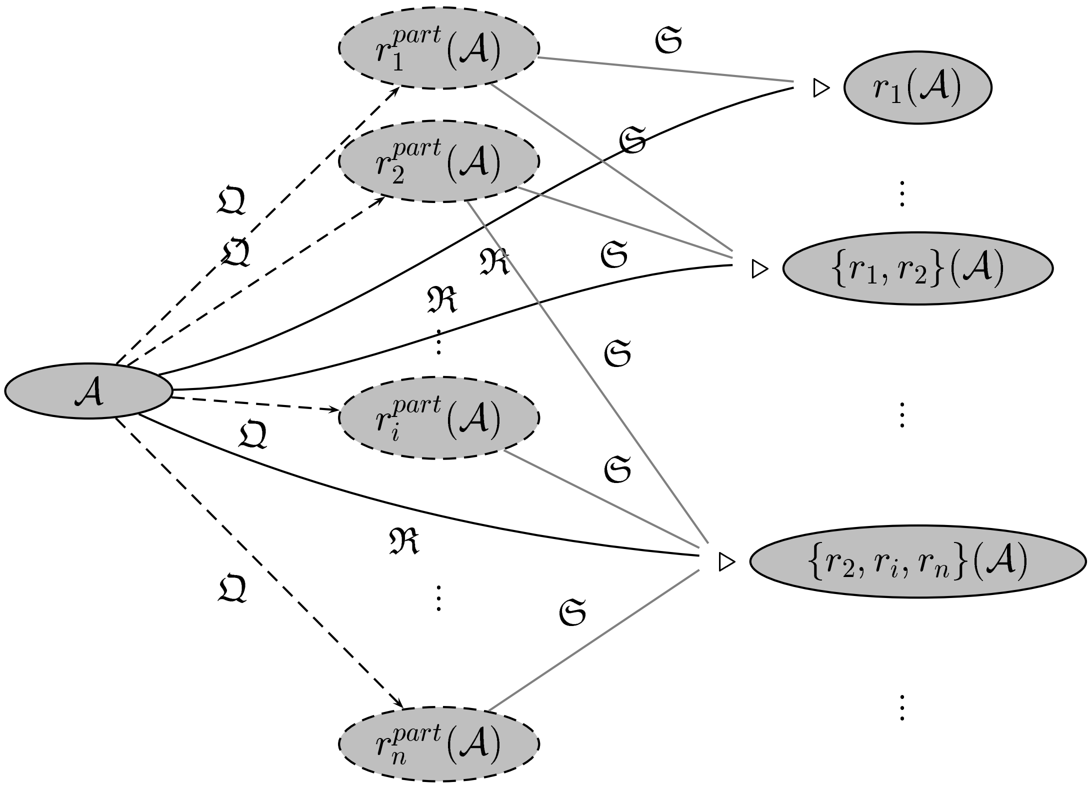


By identifying identical states it defines a directed graph consisting of all reachable states:



⇒ Kripke-Structure modeling all possible runs

Integration of the auxiliary partial interpretations:



$r^{part}(\mathcal{A})$ denotes the partial interpretation consisting of all updates done by rule r executed on $\mathcal{A}$: $r^{part}(\mathcal{A}) = r(\mathcal{A})|_{\mathsf{writes}(\mathcal{A},r)}$

$\overset{\mathfrak{Q}}{\dashrightarrow}$: Elementary accessibility of partial interpretations reached through execution of a *single* rule.

$\xrightarrow{\mathfrak{R}}$: Accessibility of successor states.

$\xrightarrow{\mathfrak{S}}$: Auxiliary accessibility relation.

$\triangleright$ computes the union of all (partial) $\mathfrak{S}$-inputs and fills the gaps with the values from the unique $\mathfrak{R}$-input.

Accessibilities:

$\mathfrak{Q}$, $\mathfrak{R}$ and $\mathfrak{S}$ have to be marked with additional information.

$$\mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \_) \text{ only if } \mathfrak{Q}(\mathcal{A}, r, \mathcal{C}) \quad , \quad \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B}) \wedge \mathfrak{S}(\mathcal{C}', \mathcal{A}, r, \mathcal{B}) \Rightarrow \mathcal{C} = \mathcal{C}'$$

$$\mathfrak{R}(\mathcal{A}, R, \mathcal{B}) \Leftrightarrow R = \{r \mid \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B})\}$$

Semantics

$$
\begin{array}{ll}
\mathfrak{R}(\mathcal{A}, R, \mathcal{B}) & \Leftrightarrow \ \mathcal{B} = \mathcal{A} \uplus \bigcup \{\mathcal{C} \mid \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B}) \wedge r \in R\} \\
\text{if } r \text{ deterministic:} & \Leftrightarrow \ \mathcal{B} = \mathcal{A} \uplus \bigcup \{\mathcal{C} \mid \mathfrak{Q}(\mathcal{A}, r, \mathcal{C}) \wedge r \in R\}
\end{array}
$$

$\mid\!\sim$: auxiliary relation: information which could be inherited from the predecessor state

$\vdash$: nonmonotonic derivability

Let f be an n-ary function symbol, $u, u_1, \ldots, u_n$ elements of the universe.

$$
\frac{\mathfrak{Q}(\mathcal{A}, r, \mathcal{C}) \quad , \quad r \text{ updates } f \text{ at } (u_1, \ldots, u_n) \text{ to } u}{\mathcal{C} \vdash f(u_1, \ldots, u_n) = u}
$$

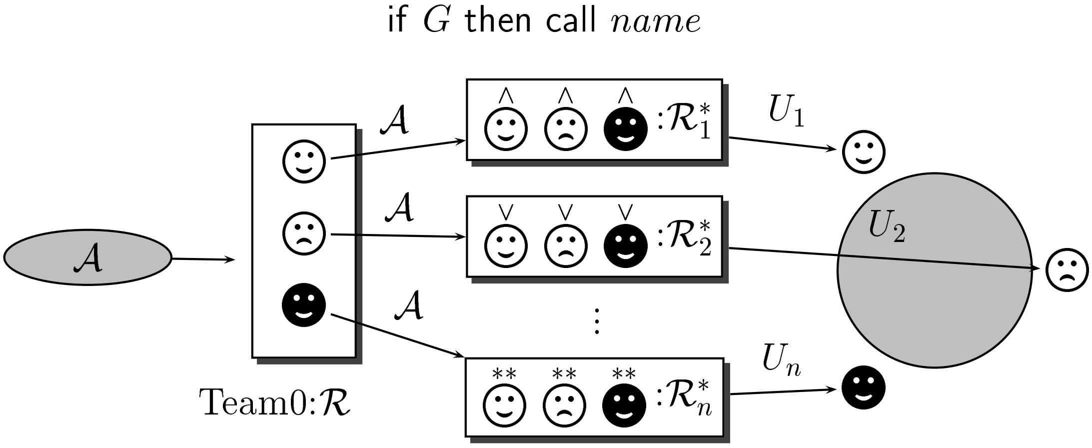
$$
\frac{\mathfrak{R}(\mathcal{A}, \_, \mathcal{B}) \quad , \quad \mathcal{A} \vdash f(u_1, \ldots, u_n) = u}{\mathcal{B} \mid\!\sim f(u_1, \ldots, u_n) = u}
$$

$$
\frac{\mathfrak{R}(\mathcal{A}, R, \mathcal{B}) \quad , \quad r \in R \quad , \quad \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B}) \quad , \quad \mathcal{C} \vdash f(u_1, \ldots, u_n) = u}{\mathcal{B} \vdash f(u_1, \ldots, u_n) = u}
$$

$$
\frac{\mathcal{A} \mid\!\sim f(u_1, \ldots, u_n) = u \quad , \quad \text{not} \ \ \mathcal{A} \vdash f(u_1, \ldots, u_n) = v \neq u}{\mathcal{A} \vdash f(u_1, \ldots, u_n) = u}
$$

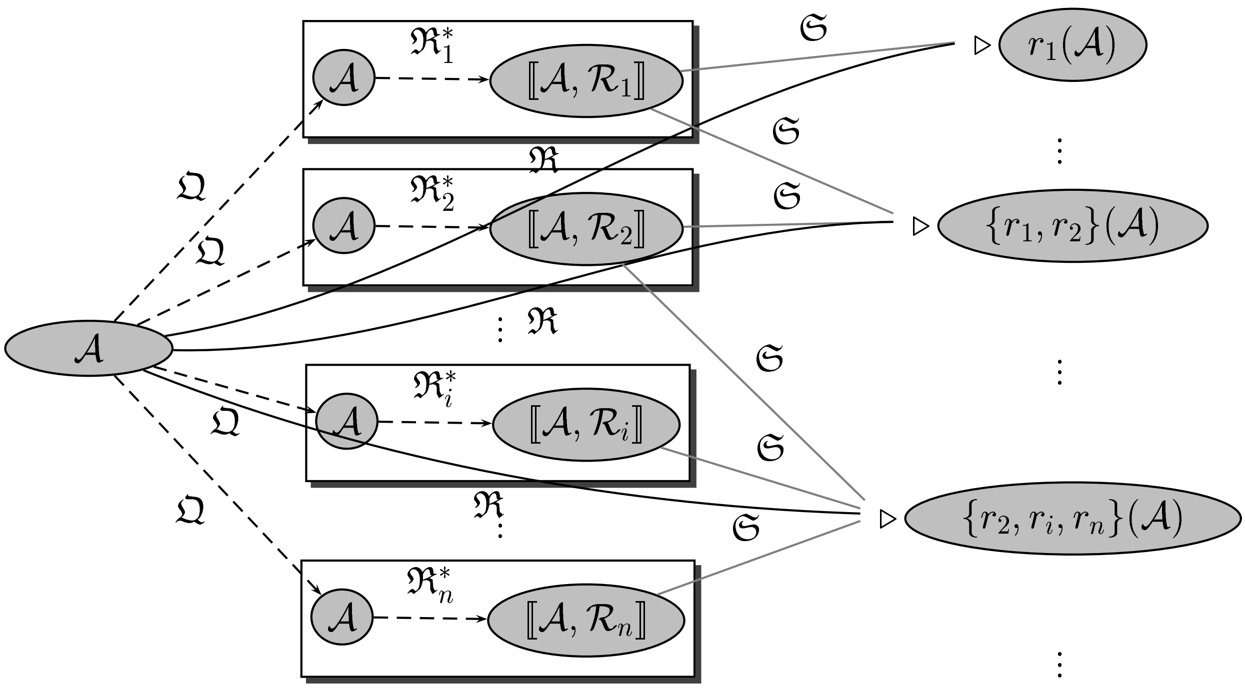
Complex Computations for Auxiliary States: Call an EA Instead of a Rule

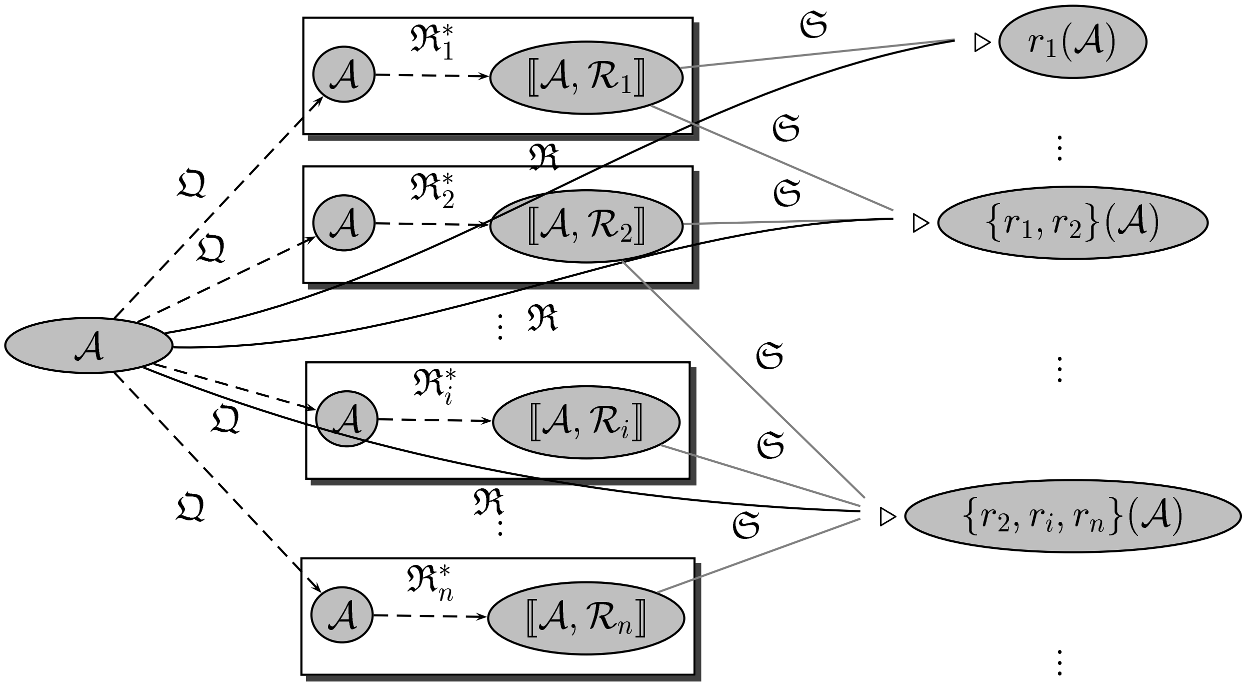
Given: EA-rule sets $\mathcal{R}_1, \mathcal{R}_2, \ldots$ – resp. different teams of demons:



where U_i is the accumulated set of updates from running the EA $(\mathcal{A}, \mathcal{R}_i)$ until a fixpoint, $[\![\mathcal{A}, \mathcal{R}_i]\!]$, is reached.

Then a hierarchically structured Kripke Model is achieved:





Instead of partial interpretations, interpretations with two qualities of truth (local knowledge and inherited knowledge) are used:

1.: Facts which are derived in the current subEA: “Procedure knowledge”.

2.: the whole interpretation “as is”; Procedure knowledge complemented by inherited information from the calling state.

Conditions on Accessibilities:

- $\mathfrak{Q}(\mathcal{A}, r, \_)$ only if r is applicable in $\mathcal{A}$
- if r is an update-rule:
  $\mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \_)$ only if $\mathfrak{Q}(\mathcal{A}, r, \mathcal{C})$
- if $r \equiv$ call $\mathcal{R}_i$ for a rule set $\mathcal{R}_i$:
  $\mathfrak{S}(\mathcal{D}, \mathcal{A}, r, \_)$ only if $\mathfrak{Q}(\mathcal{A}, r, \mathcal{C})$ and $\mathfrak{R}_i^*(\mathcal{C}, \_, \mathcal{D})$ and
  $\mathcal{R}_i$ cannot make any changes to $\mathcal{D}$
- $\mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B}) \wedge \mathfrak{S}(\mathcal{C}', \mathcal{A}, r, \mathcal{B}) \Rightarrow \mathcal{C} = \mathcal{C}'$
- $\mathfrak{R}_i(\mathcal{A}, R, \mathcal{B}) \;\Leftrightarrow\; R = \{r \mid \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B})\} \subseteq \mathcal{R}_i$

Axiomatization

$\vdash$: Facts which are derived in the current subEA: “Procedure knowledge”.

$\models$: the whole interpretation “as is”; Procedure knowledge complemented by inherited information from the calling state.

Auxiliary:

$\mathrel{|\!\!\sim}$: Procedure knowledge of the predecessor state.

$\mathrel{\vdash\!\!\bullet}$: Frame knowledge inherited from the calling state.

$$\vdash \quad = \quad \mathrel{|\!\!\sim} \uplus \text{Updates} \quad , \quad \models \quad = \quad \mathrel{\vdash\!\!\bullet} \uplus \mathrel{|\!\!\sim} \uplus \text{Updates}$$

$$\frac{\mathfrak{R}(\mathcal{A}, \_, \mathcal{B}) \quad , \quad \mathcal{A} \mathrel{\vdash\!\!\bullet} f(u_1, \ldots, u_n) = u}{\mathcal{B} \mathrel{\vdash\!\!\bullet} f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathfrak{R}(\mathcal{A}, \_, \mathcal{B}) \quad , \quad \mathcal{A} \vdash f(u_1, \ldots, u_n) = u}{\mathcal{B} \mathrel{|\!\!\sim} f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathfrak{Q}(\mathcal{A}, \_, \mathcal{B}) \quad , \quad \mathcal{A} \models f(u_1, \ldots, u_n) = u}{\mathcal{B} \mathrel{\vdash\!\!\bullet} f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathfrak{Q}(\mathcal{A}, r, \mathcal{B}) \quad , \quad r \text{ updates } f \text{ at } (u_1, \ldots, u_n) \text{ to } u}{\mathcal{B} \vdash f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathfrak{R}(\mathcal{A}, R, \mathcal{B}) \quad , \quad r \in R \quad , \quad \mathfrak{S}(\mathcal{C}, \mathcal{A}, r, \mathcal{B}) \quad , \quad \mathcal{C} \vdash f(u_1, \ldots, u_n) = u}{\mathcal{B} \vdash f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathcal{A} \vdash f(u_1, \ldots, u_n) = u}{\mathcal{A} \models f(u_1, \ldots, u_n) = u}$$

$$\frac{\mathcal{A} \mathrel{|\!\!\sim} f(u_1, \ldots, u_n) = u \quad , \quad \text{not} \ \ \mathcal{A} \vdash f(u_1, \ldots, u_n) = v \neq u}{\mathcal{A} \vdash f(u_1, \ldots, u_n) = u}$$

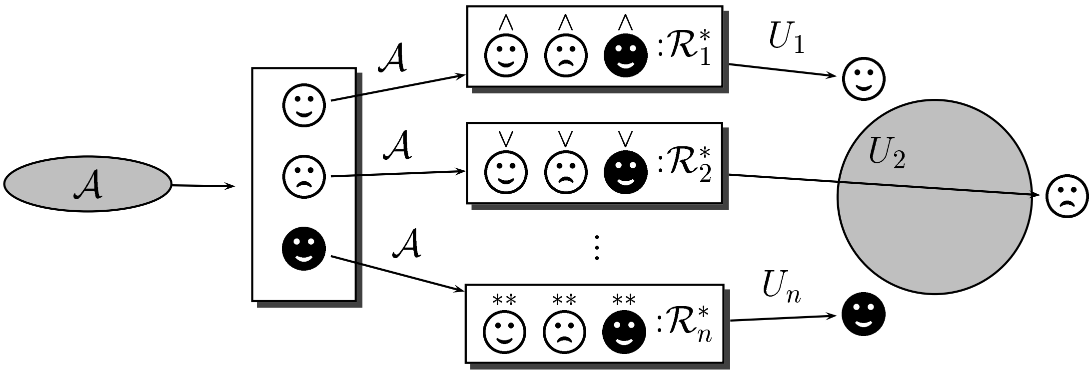
$$\frac{\mathcal{A} \mathrel{\vdash\!\!\bullet} f(u_1, \ldots, u_n) = u \quad , \quad \text{not} \ \ \mathcal{A} \vdash f(u_1, \ldots, u_n) = v \neq u}{\mathcal{A} \models f(u_1, \ldots, u_n) = u}$$

One more step further: Distributed Structured Programming with EA's

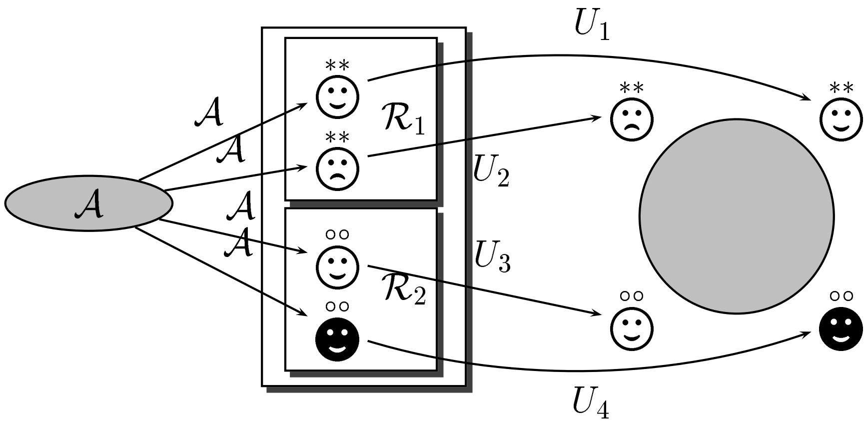
A "state" can be given at any position to any other team of demons – representing a different set of rules.

Two types of parallel composition

Encapsulated Parallel: Transitions: $\mathcal{R}_1^* \cup \mathcal{R}_2^* \cup \mathcal{R}_3^*$



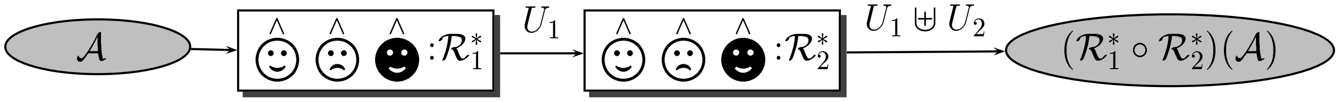
Joined parallel: Transitions: $\mathcal{R}_1 \cup \mathcal{R}_2$



Sequential Composition

$$\textsf{if } G \textsf{ then call } name_1 \;;\; \textsf{call } name_2 \;; \ldots ;\; \textsf{call } name_i$$

Encapsulated Sequential: Transitions: $(\mathcal{R}_1^*) \circ (\mathcal{R}_2^*)$



Specification of Processes by Systems of EAs

- Specifying semantically closed subsystems
- Combining them

$$
\begin{aligned}
(\mathcal{INIT}) \circ ((\mathcal{CO}\text{mmunications})^* \circ\ & ((\mathcal{T}\text{ravel Agency})^* \cup \\
& (\mathcal{CA}\text{rrier})^* \cup \\
& (\mathcal{R}\text{ent-a-Car})^* \cup \\
& (\mathcal{C}\text{lient}_1)^* \cup \ldots \cup\ (\mathcal{C}\text{lient}_2)^*))^*
\end{aligned}
$$

Dining Philosophers:

$$(\mathcal{S}\text{et up } \mathcal{T}\text{able}) \circ (\mathcal{P}\text{hil}_1 \cup \ldots \cup \mathcal{P}\text{hil}_n)^*$$

A related concept has been worked out for a state- and transition-oriented language for active databases.